ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Інтегрований веб-ресурс із впровадженням інтелектуального рекламного помічника на базі React/Vue/TypeScript»

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 Комп'ютерні науки

(код, найменування спеціальності) освітньо-професійної програми Штучний інтелект

(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Костянтин ЦІКРА

(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: Костянтин ЦІКРА
здобувач вищої освіти гр.ШІД-41

Керівник: Андрій БОНДАРЧУК
д.т.н, професор

Рецензент: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Штучного інтелекту
Ступінь вищої освіти Бакалавр
Спеціальність 122 Комп'ютерні науки
Освітньо-професійна програма Штучний інтелект

ЗАТВЕРДЖУЮ
Завідувач кафедрою Штучного інтелекту

_____ Ольга ЗІНЧЕНКО
«_____» _____ 2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Цікрі Костянтину Сергійовичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Інтегрований веб-ресурс із впровадженням інтелектуального рекламного помічника на базі React/Vue/TypeScript
керівник кваліфікаційної роботи Андрій БОНДАРЧУК д.т.н., професор
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» 02.2025 р. № 56

2. Строк подання кваліфікаційної роботи «23» травня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: параметри інтелектуальної веб-платформи, вимоги до автоматизації PR-послуг, сучасні веб-технології, моделі штучного інтелекту.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): дослідження PR-послуг; розробка програмного продукту та демонстрація результатів; аналіз наявної науково-технічної літератури.

5. Перелік графічного матеріалу: презентація, опис процесу роботи програми, архітектура веб-платформи PR-послуг, опис використаних інструментів, висновки

6. Дата видачі завдання «24» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз джерел з теми цифрового PR та сучасних веб-технологій	21.04-25.04.25	Виконано
2	Вивчення функціональних вимог до платформи	26.04-30.04.25	Виконано
3	Проєктування архітектури системи та структури бази даних	01.05-06.05.25	Виконано
4	Розробка клієнтської частини (інтерфейс, взаємодія з API, UX/UI дизайн)	07.05-13.05.25	Виконано
5	Розробка серверної частини та REST API	14.05-18.05.25	Виконано
6	Інтеграція інтелектуального помічника та чат-функціоналу	19.05-22.05.25	Виконано
7	Тестування системи, усунення помилок та оптимізація	23.05-27.05.25	Виконано
8	Підготовка та розробка демонстраційних матеріалів	28.05-29.05.25	Виконано
8	Оформлення роботи: вступ, висновки, реферат	29.05-30.05.25	Виконано

Здобувач вищої освіти _____ Костянтин ЦІКРА
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Андрій БОНДАРЧУК
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 68 стор., 1 табл., 15 рис., 20 джерел.

Мета роботи — розробити інтегрований веб-ресурс для оптимізації рекламної діяльності з інтелектуальним помічником, використовуючи сучасні веб-технології та штучний інтелект.

Об'єкт дослідження — процеси цифрової реклами, маркетингових комунікацій та автоматизації рекламної діяльності.

Предмет дослідження — методи створення односторінкових веб-застосунків (SPA) на базі React/Vue/TypeScript; застосування штучного інтелекту для генерації рекламних рекомендацій, оптимізації кампаній та обробки запитів користувачів.

Короткий зміст роботи: У роботі проаналізовано сучасні підходи до побудови інтерактивних веб-ресурсів, зокрема застосування фреймворків React/Vue, мови TypeScript та хмарних баз даних.

В першому розділі **проаналізовано предметну область дослідження**, розглянуто сучасні тенденції в цифровій рекламі та маркетингу, виявлено проблеми, які можуть бути вирішені за допомогою інтелектуальних систем.

В другому розділі **спроектовано архітектуру інтегрованого веб-ресурсу**. Деталізовано структуру клієнтської частини на React/Vue з використанням TypeScript, архітектуру бекенду та обраної бази даних. Розглянуто аспекти UX/UI-дизайну та забезпечення безпеки даних.

В третьому розділі **реалізовано основні функціональні модулі веб-ресурсу**. Описано налаштування середовища розробки, розробку інтерфейсу користувача, інтеграцію з обраними API для ШІ-моделі та бази даних, а також функціонал, пов'язаний з генерацією рекламного контенту та рекомендацій.

КЛЮЧОВІ СЛОВА: веб-ресурс, реклама, маркетинг, інтелектуальний помічник, штучний інтелект, React, Vue, TypeScript, GPT, оптимізація рекламних кампаній, SPA.

ABSTRACT

The text of the bachelor's qualification paper: 68 pages, 15 figures, 1 table, 20 sources.

The aim of the work is to develop an integrated web resource for optimizing advertising activities with an intelligent assistant, using modern web technologies and artificial intelligence.

The object of research is the processes of digital advertising, marketing communications, and the automation of advertising activities.

The subject of research includes methods for creating single-page web applications (SPA) based on React/Vue/TypeScript; the application of artificial intelligence for generating advertising recommendations, campaign optimization, and processing user requests.

Brief content of the work: The paper analyzes modern approaches to building interactive web resources, particularly the use of React/Vue frameworks, TypeScript language, and cloud databases.

The first chapter analyzes the research domain, examines current trends in digital advertising and marketing, and identifies problems that can be solved using intelligent systems.

The second chapter designs the architecture of the integrated web resource. The structure of the client-side using React/Vue with TypeScript, the backend architecture, and the chosen database are detailed. Aspects of UX/UI design and data security are considered.

The third chapter implements the main functional modules of the web resource. The development environment setup, user interface development, integration with selected APIs for the AI model and database, as well as functionality related to advertising content generation and recommendations, are described.

KEYWORDS: web resource, advertising, marketing, intelligent assistant, artificial intelligence, React, Vue, TypeScript, GPT, advertising campaign optimization, SPA.

ЗМІСТ

ВСТУП

У сучасному цифровому світі комунікаційні процеси між брендами, бізнесами та їхньою аудиторією зазнають кардинальних змін, зумовлених швидким розвитком інформаційних технологій. Паблік рилейшнз (PR) як ключовий елемент стратегічного менеджменту виходить за рамки традиційних підходів, інтегруючись у цифрові екосистеми, де пріоритетами є оперативність, персоналізація та автоматизація. У цьому контексті зростає потреба в інноваційних веб-платформах, які здатні оптимізувати PR-процеси, забезпечувати миттєву взаємодію з клієнтами та адаптувати стратегії до динамічних ринкових умов.

Актуальність теми дослідження зумовлена необхідністю створення універсальних цифрових інструментів, які б відповідали запитам малого та середнього бізнесу, фрилансерів і PR-агенцій. Такі інструменти мають поєднувати зручність використання, технологічну гнучкість і можливості інтеграції з сучасними сервісами, щоб забезпечити ефективне управління комунікаціями без значних ресурсних витрат. Розробка односторінкової веб-платформи (SPA) на основі передових технологій, таких як React, TypeScript і алгоритми штучного інтелекту, є відповіддю на ці виклики.

Об'єктом дослідження є процес автоматизації PR-комунікацій у цифровому середовищі.

Предметом дослідження виступає система веб-платформи для автоматизованого управління PR-процесами, що включає інтелектуальні рекомендації, аналітику, інтеграцію з зовнішніми сервісами та модулі для аналізу ефективності кампаній.

Метою роботи є створення та впровадження SPA-платформи для оптимізації PR-діяльності, яка поєднує сучасні технології веб-розробки з елементами штучного інтелекту.

Завданнями дослідження є:

- Вивчення сучасних підходів до автоматизації PR-процесів та аналіз їхніх переваг і недоліків.

- Проєктування та розробка веб-платформи з інтуїтивним інтерфейсом і підтримкою реального часу.
- Інтеграція інтелектуального помічника для генерації персоналізованих PR-рекомендацій.
- Забезпечення безпеки даних користувачів через шифрування та захищену авторизацію.
- Тестування платформи та оцінка її ефективності в реальних умовах.
- Розробка модуля для автоматичного аналізу ефективності PR-кампаній на основі метрик залученості аудиторії.
- Інтеграція платформи з популярними платіжними системами для підтримки монетизації сервісів.

Методика дослідження базується на застосуванні методів веб-розробки, включаючи проєктування односторінкових додатків, використанні фреймворків React і TypeScript, а також інтеграції WebSocket для реального часу. Для створення інтелектуального помічника використовуються методи обробки даних і машинного навчання. Додатково застосовуються принципи безпечного програмування для захисту даних і тестування користувацького досвіду (UX/UI) для забезпечення зручності платформи.

Дослідження базується на аналізі відкритих джерел, документації сучасних фреймворків (React, Vue), бібліотек для роботи з WebSocket (Socket.io), а також наборів даних для навчання моделей машинного навчання. Додатково використовуються стандарти захисту даних (наприклад, GDPR) і приклади інтеграції з платіжними системами та аналітичними сервісами. Наукова новизна роботи полягає в розробці комплексної SPA-платформи, яка інтегрує сучасні веб-технології з алгоритмами штучного інтелекту для автоматизації PR-процесів. Особливістю є акцент на підтримці малого та середнього бізнесу, а також забезпечення високого рівня безпеки та адаптивності платформи до ринкових змін.

Результати дослідження можуть бути застосовані в PR-агенціях, маркетингових відділах компаній, а також фрилансерами для створення

ефективних комунікаційних стратегій. Платформа дозволить автоматизувати рутинні процеси, аналізувати ефективність кампаній у реальному часі та генерувати персоналізовані рекомендації. У перспективі розробка може бути масштабована для інтеграції з CRM-системами, розширення аналітичних можливостей і вдосконалення інтелектуальних функцій.

1 ТЕОРЕТИЧНІ ОСНОВИ СТВОРЕННЯ ІНТЕЛЕКТУАЛЬНИХ ВЕБ-ПЛАТФОРМ PR-ПОСЛУГ

1.1 Роль PR у цифровому середовищі

Сфера зв’язків із громадськістю (PR) зазнала суттєвих трансформацій під впливом цифрових технологій. Якщо раніше PR асоціювався переважно з прес-релізами, медіа-подіями та офлайн-активностями, то сьогодні він інтегрувався в онлайн-простір, де основними інструментами стали соціальні мережі, цифрові платформи, контент-маркетинг та аналітика даних.

У цифровому середовищі PR не просто виконує інформаційну функцію, а стає ключовим чинником формування довіри до бренду, взаємодії з аудиторією в режимі реального часу, керування репутаційними ризиками та стимулювання лояльності. Основна особливість такого середовища — це відкритість, швидкість поширення інформації та багатоканальність комунікацій. Користувачі більше не є пасивними споживачами новин — вони беруть активну участь у формуванні контенту, реагують, коментують, поширюють інформацію. Це створює нову реальність, у якій PR-фахівець має діяти не лише як комунікатор, а як стратег, аналітик і модератор цифрової взаємодії.

Цифровий PR зазнав значних змін завдяки впровадженню інструментів моніторингу та аналітики, які дозволяють відстежувати ефективність кампаній, тональність контенту та рівень залученості аудиторії в реальному часі. Платформи, такі як Google Analytics, Brandwatch, Mention і Sprout Social, надають PR-фахівцям можливість оперативно реагувати на зовнішні виклики, адаптуючи комунікаційні стратегії. Ці інструменти дозволяють відійти від ретроспективного аналізу, забезпечуючи миттєве коригування повідомлень і вибір оптимальних каналів поширення [1].

Ще одним фундаментальним зсувом є глобалізація інформаційного простору. Цифрове середовище не знає географічних меж — повідомлення бренду можуть миттєво поширюватися на будь-яку аудиторію, незалежно від її

місцезнаходження. У результаті зростає потреба в локалізації контенту, врахуванні культурних та мовних особливостей, а також в управлінні репутацією на міжнародному рівні.

Особливе місце у цифровому PR займають соціальні мережі, які не лише стали основним каналом комунікації, а й створили унікальні можливості для таргетованого впливу. Сьогодні компанії не просто публікують інформацію — вони ведуть діалог із користувачами, формують спільноти, залучають лідерів думок та впливають на формування громадської думки. Саме тут PR-послуги дедалі частіше автоматизуються: наприклад, чат-боти в месенджерах забезпечують первинну комунікацію з клієнтами, AI-алгоритми прогнозують реакції на повідомлення, а інструменти штучного інтелекту формують рекомендації щодо контенту.

У сфері цифрового PR дедалі більшої ваги набувають інтелектуальні технології. Завдяки алгоритмам машинного навчання, обробці природної мови (NLP) і системам рекомендацій PR-фахівці можуть краще розуміти поведінкові патерни аудиторії, прогнозувати її реакції та розробляти персоналізовані комунікаційні підходи. Наприклад, платформи на базі штучного інтелекту здатні аналізувати тисячі коментарів у соціальних мережах, ідентифікувати проблемні зони у сприйнятті бренду та надавати рекомендації для PR-стратегій.

Роль PR-спеціаліста в цифровому середовищі значно розширилася. Окрім класичних завдань, таких як створення прес-релізів чи організація подій, фахівці тепер займаються аналізом даних, управлінням онлайн-кампаніями, пошуковою оптимізацією (SEO), співпрацею з блогерами та протидією інформаційним ризикам, наприклад фейковим новинам чи витокам даних.

Отже, цифровий PR перетворюється на комплексну дисципліну, що інтегрує технології, комунікації та стратегічний менеджмент. Веб-платформи, які об’єднують інструменти автоматизації, аналітики та персоналізації, стають критично важливими для реалізації сучасних PR-кампаній [2].

1.2 Сучасні підходи до автоматизації PR-послуг

У контексті цифрової трансформації комунікаційних стратегій усе більше PR-агенцій, брендів та фахівців звертаються до автоматизації як до засобу підвищення ефективності, зменшення рутинного навантаження та оперативнішого реагування на інформаційні виклики. Автоматизація PR-послуг — це не просто використання програмного забезпечення, а концепція, що охоплює інтеграцію технологічних рішень на всіх етапах комунікаційного циклу: від моніторингу медіа до персоналізованої доставки контенту.

Одним із ключових напрямів автоматизації в PR є медіа-моніторинг. Такі платформи, як Meltwater, Brand24, Talkwalker і Semantrum, забезпечують відстеження згадок бренду, ключових слів, конкурентів чи тем у медіа-просторі в реальному часі. Вони автоматично оцінюють тональність контенту, аналізують джерела поширення, охоплення та географічний розподіл інформації, що дозволяє PR-фахівцям швидко реагувати на кризові ситуації або розробляти стратегічні звіти.

Ще одним важливим інструментом є автоматизація управління контентом. За допомогою CRM- і CMS-систем, таких як HubSpot, Zoho або Salesforce, фахівці можуть планувати публікації, сегментувати контент для різних аудиторій, організовувати розсилки та оцінювати ефективність кампаній. Інтеграція з платформами для управління соціальними мережами, наприклад Buffer, Hootsuite чи Sprinklr, дає змогу централізовано керувати акаунтами, автоматизувати розміщення контенту та аналізувати зворотний зв’язок.

Значну роль відіграє автоматизація email-маркетингу. Сервіси, такі як Mailchimp, Sendinblue і GetResponse, дозволяють створювати персоналізовані розсилки, сегментувати аудиторії, налаштовувати автоматичні воронки продажів і проводити A/B-тестування, що сприяє підвищенню залученості та ефективності комунікацій [3].

Автоматизовані чат-боти стають усе популярнішими в PR-діяльності, особливо для первинної комунікації, відповіді на часті запитання або збору

контактів. Вони можуть працювати на базі Facebook Messenger, Telegram або через інтеграцію з веб-сайтами. Застосування чат-ботів дозволяє не лише скоротити час відповіді, а й підвищити лояльність аудиторії через миттєву реакцію на запити.

Ще одним прогресивним напрямом є автоматизована генерація та персоналізація PR-пропозицій. Тут на допомогу приходять системи штучного інтелекту, зокрема технології NLP (Natural Language Processing) та ML (Machine Learning), які дозволяють аналізувати великі обсяги даних і формувати релевантні комунікаційні сценарії. Такі рішення здатні автоматично створювати текстові повідомлення (наприклад, новини, прес-релізи, відповіді в чаті), адаптуючи їх під конкретну цільову аудиторію, час доби або поведінкові патерни користувачів.

Автоматизація також активно застосовується в аналітиці PR-кампаній. Завдяки інтеграції з інструментами Google Analytics, Facebook Pixel, Power BI або Tableau, можна в режимі реального часу аналізувати ефективність заходів: від кількості переглядів до рівня конверсій, CTR, ER (engagement rate) та ROI кампаній. Такі дані дають змогу швидко виявляти слабкі місця, змінювати стратегії і приймати обґрунтовані рішення.

Важливу роль у цифровому PR відіграють сегментація та таргетинг, які забезпечують точне спрямування комунікаційних повідомлень до відповідних аудиторій. Сучасні алгоритми дозволяють динамічно адаптувати цільові групи на основі їхньої поведінки, інтересів, географічного розташування чи попередньої взаємодії з брендом. Завдяки цьому кожен користувач отримує персоналізовану та релевантну інформацію, що значно підвищує шанси на залучення та зворотний зв’язок [4].

Підсумовуючи, можна стверджувати, що автоматизація PR-послуг — це не лише технологічний тренд, а необхідність у сучасних умовах. Вона дозволяє зекономити час, знизити витрати, покращити якість комунікації та підвищити контроль над інформаційними потоками. Водночас, важливо зберігати баланс між автоматизацією та людською емпатією, яка залишається серцем кожної успішної PR-стратегії.

1.3 Огляд технологій для побудови SPA (React, Vue, TypeScript)

Із розвитком цифрових сервісів та зростанням вимог до зручності користування, зокрема в сфері PR-платформ, архітектура Single Page Application (SPA) стала стандартом для багатьох сучасних веб-застосунків. Її головна перевага — це безперервність користувацького досвіду: замість перезавантаження сторінки під час кожної дії користувача, SPA динамічно оновлює лише необхідний вміст, що значно покращує швидкість, зручність і реактивність системи.

PR-платформи, що використовують SPA, дозволяють створювати адаптивні інтерфейси для редакторів контенту, аналітиків та клієнтів у єдиному середовищі — без потреби переходу між сторінками чи довгого очікування завантаження. Такий підхід особливо актуальний, коли мова йде про динамічний контент (звітність, чат, інструменти персоналізації), яким часто користуються представники PR-агенцій.

Серед інструментів для розробки односторінкових веб-додатків (SPA) особливе місце посідають React, Vue і TypeScript, які поєднують високу продуктивність, масштабованість і широку підтримку спільноти розробників. Розглянемо їхні особливості.

React.js, JavaScript-бібліотека, створена компанією Facebook, є однією з провідних технологій для побудови інтерфейсів користувача. Її популярність зумовлена гнучкістю, розвиненою екосистемою та компонентним підходом, який дозволяє створювати ізольовані та повторно використовувані елементи інтерфейсу [5]. Серед переваг React варто виокремити:

- Віртуальний DOM, що забезпечує швидке оновлення інтерфейсу без повного перерендеру сторінки;
- JSX-синтаксис, який дозволяє писати HTML-подібний код у JavaScript, роблячи розробку зручнішою;
- Широка спільнота та набір бібліотек (Redux, React Router, Next.js), які значно розширюють функціональність.

У контексті розробки PR-платформ React відіграє ключову роль завдяки своїй здатності створювати динамічні та зручні інтерфейси користувача. Ця JavaScript-бібліотека дозволяє реалізовувати інтерактивні дашборди для відображення аналітичних даних, таких як метрики залученості аудиторії чи ефективність кампаній, а також формувати динамічні списки контактів із можливістю фільтрації та сортування. Крім того, React підтримує створення адаптивних форм для генерації PR-пропозицій, які автоматично оновлюються залежно від введених даних, що значно спрощує взаємодію користувачів із платформою. Компонентний підхід React забезпечує модульність, дозволяючи розробникам створювати ізольовані елементи інтерфейсу, які легко масштабувати та підтримувати в умовах складних PR-систем [6].

Vue.js, своєю чергою, є прогресивним JavaScript-фреймворком, який часто порівнюють із React за гнучкістю, але вирізняється простотою освоєння. Завдяки зрозумілому шаблонному синтаксису, детальній документації та підтримці двостороннього зв’язування даних Vue ідеально підходить для створення інтерактивних компонентів, таких як чати для комунікації з клієнтами, форми для збору зворотного зв’язку чи модулі для управління розсилками. Двостороннє зв’язування даних спрощує синхронізацію між моделлю даних і відображенням, що особливо корисно для реального часу в PR-додатках, наприклад, під час моніторингу коментарів у соціальних мережах. Vue також підтримує поступову інтеграцію, що дозволяє додавати його до наявних проєктів без необхідності повної переробки коду. Обидва фреймворки, React і Vue, завдяки своїм можливостям значно підвищують ефективність розробки PR-платформ, забезпечуючи швидке створення гнучких і користувацько-орієнтованих інтерфейсів [7].

Переваги Vue:

1. Легкий старт та хороша підтримка для малих і середніх проєктів;
2. Вбудовані інструменти, зокрема Vue Router та Vuex для маршрутизації та керування станом;
3. Висока продуктивність і гнучкість завдяки модульній структурі.

Для SPA-платформи у сфері PR Vue чудово підходить, якщо команда розробників невелика або потрібно швидко розгорнути MVP (мінімально життєздатний продукт) із можливістю подальшого масштабування.

TypeScript, розроблений компанією Microsoft, є надмножиною JavaScript, що впроваджує статичну типізацію. Ця технологія стала стандартом для масштабних веб-проєктів завдяки підвищеній стабільності коду та передбачуваності, що особливо цінно для великих команд і проєктів із тривалим циклом розробки [8].

Розглянемо переваги TypeScript у SPA:

- Покращує читабельність і надійність коду завдяки системі типів;
- Зменшує кількість помилок під час виконання (runtime errors), попереджаючи їх на етапі компіляції;
- Полегшує роботу з API, що є важливою частиною будь-якої PR-платформи, особливо з точки зору інтеграцій із зовнішніми сервісами (аналітика, платежі, CRM).

Завдяки інтеграції з React або Vue, TypeScript дозволяє створювати надійні SPA-додатки з чіткою структурою та зручним автодоповненням коду, що пришвидшує розробку.

1.4 Інтелектуальні помічники в рекламі: алгоритми, тренди та інструменти

Із розвитком штучного інтелекту (ШІ) цифровий PR і рекламна індустрія зазнали кардинальних змін. Сьогодні інтелектуальні помічники стали невіддільною частиною ефективних рекламних стратегій: вони аналізують поведінку споживачів, персоналізують контент, оптимізують бюджет кампаній і навіть ведуть комунікацію з клієнтами. У сфері PR це означає новий рівень автоматизації, глибшого розуміння цільової аудиторії та побудову ефективних персоналізованих повідомлень.

Інтелектуальні помічники являють собою програмні системи, які використовують технології штучного інтелекту, зокрема алгоритми машинного навчання (ML), обробку природної мови (NLP), аналіз великих даних (Big Data) та інші інструменти, для автоматизації завдань, що раніше вимагали людського втручання. Їхня основна перевага полягає в здатності оперативно обробляти значні обсяги даних, аналізувати історію взаємодії користувачів, їхні запити, інтереси та навіть емоційні реакції, щоб надавати персоналізовані рішення. У контексті PR і реклами інтелектуальні помічники дозволяють оптимізувати комунікаційні стратегії, підвищувати залученість аудиторії та прогнозувати ефективність кампаній.

Серед ключових алгоритмів, які застосовуються в таких системах, можна виокремити:

1. Кластеризація (clustering) — методи, що групують користувачів за їхньою поведінкою, демографічними чи іншими характеристиками для створення персоналізованих пропозицій;
2. Рекомендаційні системи (на основі колаборативної або контентної фільтрації) — генерують релевантний контент, враховуючи вподобання користувача чи схожих аудиторій;
3. Аналіз тональності (sentiment analysis) — ідентифікує емоційне забарвлення коментарів, відгуків чи запитів, що допомагає оцінити репутацію бренду;
4. Нейронні мережі, зокрема рекурентні нейронні мережі (RNN) і трансформери, — використовуються для генерації текстів, автоматизації відповідей у чат-ботах і обробки природної мови;
5. Прогнозна аналітика (predictive analytics) — передбачає ймовірні реакції аудиторії, показники клікабельності (CTR), конверсії чи потенційні репутаційні ризики.

Ці алгоритми дозволяють інтелектуальним помічникам не лише автоматизувати рутинні процеси, а й створювати адаптивні стратегії, що відповідають динамічним потребам аудиторії та бізнесу [9].

Усе це дозволяє створити систему, яка не просто надає шаблонну інформацію, а діє як динамічний учасник комунікації, здатний підлаштовуватися під контекст і потреби користувача.

На практиці інтелектуальні помічники реалізуються за допомогою таких інструментів:

- ChatGPT, Bard, Claude AI — генерація текстів, відповідей, рекомендацій на основі діалогу з користувачем;
- Persado, Jasper.ai, Copy.ai — створення рекламних текстів на базі психологічного аналізу аудиторії;
- Drift, Intercom, Tidio — інтелектуальні чат-платформи, які аналізують запити клієнтів та автоматизують спілкування;
- HubSpot AI, Salesforce Einstein — рекомендаційні механізми в CRM-системах;
- Facebook Ads AI, Google Ads Smart Campaigns — інтелектуальне розміщення реклами з урахуванням прогнозованих дій користувача.

Дані інструменти дозволяють брендам оперативно адаптувати контент, керувати бюджетами, оцінювати результати кампаній у реальному часі та мінімізувати людський фактор. Серед ключових трендів у розвитку інтелектуальних помічників варто виділити:

1. *Гіперперсоналізацію* — відображення контенту, створеного спеціально для конкретного користувача на основі його інтересів, історії переглядів, геолокації тощо;
2. *Голосові інтерфейси* — розширення каналів взаємодії з користувачем за допомогою голосових помічників (Alexa, Google Assistant);
3. AI-креативність — генерація візуального і текстового контенту за запитом (midjourney, DALL·E, GPT-4);
4. Інтеграція з мультимодальними інструментами — обробка тексту, зображення, відео, аналітики у єдиній системі.

У контексті дипломної роботи інтелектуальний помічник буде містити наступні функції:

- Аналіз запитів користувача: виявлення намірів, тематики, стилістичних переваг;
- Формування PR-рекомендацій: вибір релевантних стратегій, заголовків, каналів комунікації;
- Персоналізація пропозицій: створення повідомлень або кампаній відповідно до профілю клієнта;
- Збір фідбеку і трендовий аналіз: оцінка реакції на кампанії, адаптація стилістики чи тематики.

Таке рішення дозволяє розробити платформу розумною: вона повинна не лише передавати повідомлення, а й сприти формуванню ефективної комунікаційної стратегії на основі даних.

В результаті, було встановлено, що роль PR у цифровому середовищі кардинально змінилася, перетворившись на інтерактивну та багатоканальну дисципліну, яка вимагає не тільки стратегічного планування, а й швидкої адаптації до динамічних змін онлайн-простору. Сучасні підходи до автоматизації PR-послуг демонструють зростаючу залежність від технологічних рішень, що дозволяють оптимізувати процеси моніторингу, аналізу даних, управління кампаніями та взаємодії з аудиторією.

Огляд технологій для побудови односторінкових додатків (SPA), зокрема React, Vue та TypeScript, показав, що ці фреймворки та мова програмування є оптимальними інструментами для створення високопродуктивних, масштабованих та зручних у користуванні веб-платформ. React та Vue пропонують ефективні механізми для розробки інтерактивного інтерфейсу, а TypeScript забезпечує типову безпеку, що є критично важливим для великих проектів та командної розробки.

Вивчення цих технологій підкреслює потенціал штучного інтелекту у персоналізації рекламних кампаній, оптимізації цільової аудиторії та автоматизації прийняття рішень, що робить їх невід'ємною частиною сучасної PR-платформи. Таким чином, інтеграція інтелектуальних алгоритмів на базі React/Vue/TypeScript є обґрунтованим кроком для створення інноваційного веб-ресурсу, який відповідатиме сучасним вимогам цифрового PR.

2 ПРОЄКТУВАННЯ ІНТЕГРОВАНОЇ ВЕБ-ПЛАТФОРМИ PR-ПОСЛУГ

2.1 Визначення вимог до системи

Розробка сучасної веб-платформи PR-послуг потребує чіткого формалізованого підходу до визначення вимог. Саме на цьому етапі формується технічне бачення продукту, встановлюється узгоджене розуміння цілей між замовником, розробниками та користувачами. Успішна реалізація платформи залежить від повноти, однозначності та обґрунтованості вимог, які мають охоплювати як функціональні можливості, так і нефункціональні характеристики — продуктивність, масштабованість, безпеку тощо.

Платформа, що розробляється в межах цієї дипломної роботи, призначена для автоматизованого надання PR-послуг, включно з генерацією персоналізованих рекомендацій, керуванням кампаніями, веденням аналітики та взаємодією з клієнтами через чат-інтерфейс. Крім того, система повинна забезпечити адміністрування акаунтів, обробку платежів, зберігання історії взаємодій і можливість розширення функціоналу.

Бізнес-вимоги до PR-платформи визначають її стратегічні цілі, відповідаючи на запитання, які проблеми продукт має вирішити для користувачів і власників бізнесу. У контексті розробки платформи основна увага приділяється забезпеченню доступу малого та середнього бізнесу до професійних PR-послуг без необхідності залучення окремої команди фахівців. Це досягається через автоматизацію ключових процесів, таких як комунікація з клієнтами за допомогою інтелектуального помічника, який використовує алгоритми штучного інтелекту для обробки запитів і надання персоналізованих рекомендацій. Платформа також передбачає створення гнучкого середовища, яке дозволяє користувачам ефективно управляти контентом, отримувати аналітичні звіти та адаптувати стратегії на основі даних у реальному часі. Важливим аспектом є інтеграція аналітичних інструментів, які надають можливість оцінювати ефективність PR-кампаній через

метрики, такі як залученість аудиторії, охоплення чи конверсії. Крім того, бізнес-модель платформи базується на монетизації через гнучкі підписки, разові послуги та інтеграцію з платіжними системами, що забезпечує фінансову стійкість і доступність для різних категорій користувачів. Такий підхід дозволяє платформі відповідати потребам сучасного ринку, де швидкість, економічність і персоналізація є ключовими факторами успіху [10].

Мета даного проєкту полягає у визначенні загального напряму функціоналу та архітектури платформи. Функціональні вимоги чітко окреслюють дії системи, операції, які вона повинна виконувати, та її реакцію на взаємодію з користувачем.

Основні функціональні блоки майбутньої платформи включатимуть систему реєстрації та авторизації користувачів, що передбачає створення різних типів акаунтів – для звичайних користувачів, адміністраторів та модераторів. Авторизація здійснюватиметься як за допомогою електронної пошти та пароля, так і через OAuth-провайдерів, таких як Google та LinkedIn, а також буде реалізовано функціонал відновлення пароля та верифікації електронної пошти.

Модуль профілю користувача дозволить редагувати контактну інформацію, керувати підписками та переглядати історію операцій. Користувачі матимуть можливість завантажувати аватари, обирати мову інтерфейсу та налаштовувати повідомлення.

Центральним елементом стане інтелектуальний помічник PR, який аналізуватиме введені запити щодо теми, типу послуги та цільової аудиторії. На основі цього аналізу помічник генеруватиме PR-рекомендації, що включатимуть текст, стратегію та канали розповсюдження, а результати будуть персоналізовані відповідно до профілю користувача.

Для управління PR-кампаніями буде реалізовано модуль, що дозволить створювати кампанії із зазначенням заголовка, опису та дедлайнів. Кампанії можна буде прив'язувати до конкретних тем, аналітичних даних та ключових повідомлень, а їхня структура буде візуалізована на дашборді.

Інтеграція з чат-модулем забезпечить можливість спілкування користувача з помічником або реальним оператором у режимі реального часу. Система

оброблятиме вхідні запити та надаватиме підказки на основі запитів, використовуючи автозаповнення та рекомендації штучного інтелекту.

Платіжна система платформи надаватиме вибір тарифних планів, підключення до платіжних систем, таких як Stripe або PayPal, а також міститиме історію платежів та функцію автоматичного виставлення рахунків.

Аналітичний модуль кампаній включатиме графіки та ключові показники ефективності (KPI), такі як охоплення, залучення та кліки. Він відображатиме історію дій користувача та клієнтів, а також матиме інтеграцію з Google Analytics та Facebook Pixel для поглибленого аналізу.

Нефункціональні вимоги визначають якість реалізації системи, охоплюючи її продуктивність, зручність, безпеку та масштабованість. Щодо продуктивності, система повинна забезпечувати завантаження основної сторінки не довше ніж за 3 секунди, підтримувати паралельну роботу щонайменше 1000 користувачів та забезпечувати оновлення даних у реальному часі без перезавантаження сторінки за допомогою WebSocket.

Масштабованість системи передбачає горизонтальне масштабування, що досягатиметься завдяки модульній архітектурі та використанню мікросервісів, а також можливості підключення додаткових API, таких як GPT, CRM та email-сервіси.

Безпека системи гарантуватиметься шифруванням персональних даних за допомогою SSL, HTTPS та bcrypt, захистом від поширених веб-атак, таких як XSS, CSRF та SQL Injection, веденням журналів активності та обмеженням прав доступу.

UX/UI модулі забезпечуватимуть адаптивний дизайн для різних типів пристроїв, сучасний інтерфейс з врахуванням принципів доступності (WCAG), легку навігацію, підказки та онбординг для нових користувачів.

Для забезпечення сумісності та ефективної командної роботи встановлені такі базові технічні обмеження: фронтенд буде розроблено на React або Vue з використанням TypeScript, бекенд – на Node.js з Express та REST API, база даних – PostgreSQL або MongoDB, хостинг – хмарні сервіси (Render, Vercel, Firebase,

AWS), а для DevOps використовуватимуться Docker, GitHub Actions для CI/CD та Grafana для моніторингу.

2.2 Архітектура клієнт-серверної взаємодії

Ефективна взаємодія між клієнтською та серверною частинами веб-платформи PR-послуг є основою її стабільної, швидкої й масштабованої роботи. У цьому проєкті обрана класична клієнт-серверна архітектура із застосуванням REST API, що забезпечує гнучкість, модульність і сумісність із зовнішніми сервісами (Fielding & Taylor, 2017). Архітектура побудована за принципом розподілу відповідальностей, де кожен компонент виконує свою чітко визначену функцію, а спілкування між ними здійснюється за допомогою стандартизованих протоколів і форматів, таких як HTTP і JSON [11].

Вся система умовно поділяється на три рівні:

1. *Клієнтська частина (Frontend)* — реалізована за допомогою React або Vue з TypeScript. Відповідає за взаємодію з користувачем, відображення інтерфейсу, динамічне оновлення контенту (Додаток А).

2. *Серверна частина (Backend)* — Node.js з Express.js. Відповідає за логіку обробки запитів, авторизацію, керування даними, логіку інтелектуального помічника, генерацію звітів (Додаток Б).

3. *База даних (Database)* — PostgreSQL (реляційна БД) або MongoDB (документо-орієнтована). Зберігає інформацію про користувачів, кампанії, повідомлення, статистику тощо (Додаток В).

Обмін даними між клієнтською і серверною частинами відбувається через HTTP/HTTPS протокол, із використанням REST API. Всі дані передаються у форматі JSON, що дозволяє легко інтегрувати зовнішні сервіси та забезпечує високий рівень сумісності. Основні типи запитів:

- GET — отримання даних (наприклад, список кампаній, профіль користувача);

- POST — створення нових об’єктів (реєстрація, нова кампанія, повідомлення);
- PUT/PATCH — оновлення існуючих даних;
- DELETE — видалення об’єктів.

На стороні клієнта реалізовано асинхронне завантаження даних за допомогою технології fetch або спеціалізованих бібліотек, таких як Axios. Це дозволяє забезпечити плавне оновлення інтерфейсу без необхідності перезавантаження сторінки, що значно покращує користувацький досвід. Для більш складних сценаріїв, що потребують миттєвої комунікації, наприклад, для реалізації сповіщень у режимі реального часу та функціоналу чату, передбачено використання WebSocket-з'єднань, які реалізуються за допомогою бібліотеки Socket.io.

Система безпеки побудована на використанні JWT (JSON Web Token) для механізму авторизації. Після успішної автентифікації сервер генерує унікальний токен, який зберігається на клієнтській стороні, зазвичай у localStorage, і додається до кожного наступного запиту як Authorization header. Для забезпечення всебічної безпеки всі запити передаються через протокол HTTPS, дані валідуються як на клієнтському, так і на серверному боці, а захист від SQL-ін'єкцій забезпечується через використання ORM або параметризованих запитів. Крім того, система підтримує відновлення сесій за допомогою refresh-токенів та реалізує механізм ролей доступу (адміністратор, користувач, гість) з відповідним обмеженням прав, що відповідає сучасним стандартам безпеки, як зазначено Bertocci (2016).

На серверній стороні застосовується архітектурний патерн MVC (Model-View-Controller) або Service-Oriented Architecture (SOA). Такий підхід дозволяє чітко розділити бізнес-логіку, обробку запитів та доступ до бази даних на окремі модулі, що істотно спрощує подальший супровід та розширення функціоналу системи.

Компоненти сервера відповідно розділені: контролери відповідають за обробку HTTP-запитів та взаємодію з сервісами; сервіси інкапсулюють основну бізнес-логіку, таку як генерація рекомендацій та аналіз кампаній; моделі описують

структуру даних та керують взаємодією з базою даних; а проміжне програмне забезпечення (middlewares) відповідає за авторизацію, логування та обробку помилок.

Для реалізації чат-модуля та інтелектуального помічника активно використовуються WebSocket-з'єднання через Socket.io. Це забезпечує миттєве отримання повідомлень та відповідей від помічника, усуваючи необхідність у постійних повторних HTTP-запитах. Такі сценарії використання WebSocket охоплюють живий чат між користувачем і службою підтримки або інтелектуальним помічником, сповіщення про поточний статус PR-кампаній, а також надання рекомендацій у реальному часі на основі активності користувача.

Архітектура проекту свідомо спроєктована з урахуванням майбутнього масштабування. Передбачається можливість розділення фронтенду та бекенду на окремі хости або сервери. На подальших етапах розвитку проекту планується впровадження мікросервісної структури, наприклад, шляхом винесення модуля аналітики в окремий незалежний сервіс. Завдяки REST-архітектурі, система дозволяє додавати нові API без порушення існуючої логіки, що забезпечує її гнучкість. Додатково передбачена інтеграція із зовнішніми сервісами, такими як CRM-системи, поштові служби та API штучного інтелекту, що буде здійснюватися через спеціальні адаптерні модулі, що гарантує розширюваність та взаємодію з широким спектром сторонніх рішень.

2.3 База даних: проєктування, схема, оптимізація

База даних (БД) є одним з ключових компонентів веб-платформи, що відповідає за зберігання, обробку та доступ до інформації. У контексті автоматизованої платформи PR-послуг база даних має забезпечити ефективне управління користувацькими акаунтами, кампаніями, повідомленнями, транзакціями, аналітичними даними та іншими сутностями системи. Високий рівень структурування, нормалізації та індексації даних є обов’язковою умовою для стабільної роботи всієї системи.

Для реалізації проєкту обрано реляційну СУБД PostgreSQL, яка поєднує гнучкість SQL, підтримку складних зв’язків між таблицями, транзакційність та високу продуктивність при правильній оптимізації. У разі потреби масштабування або додаткових потреб в обробці неструктурованих даних може бути розгорнута комбінована архітектура з додатковим використанням MongoDB для логів або AI-запитів.

При проєктуванні бази даних було розроблено деталізовану схему, що передбачає кілька основних сутностей, між якими встановлені взаємозв'язки типу "один-до-багатьох", "багато-до-багатьох" та "один-до-одного", відповідно до бізнес-логіки системи. Користувачі представлені сутністю "Users", яка містить унікальний ідентифікатор, електронну пошту, хеш пароля, роль, ім'я, дату створення та статус активності. З кожним користувачем пов'язані багато кампаній, транзакцій та сесій, що відображає його активність у системі.

Сутність "Campaigns" включає ідентифікатор кампанії, посилання на користувача-власника, заголовок, опис, статус, дати початку та завершення. Кожна кампанія належить одному користувачеві та може містити багато повідомлень у чаті та аналітичних записів. Повідомлення, що є частиною сутності "Messages", зберігають ідентифікатор, посилання на кампанію, тип відправника (користувач, помічник, система), зміст та дату створення, забезпечуючи діалог з інтелектуальним помічником або менеджером.

Фінансові операції користувача реєструються у сутності "Payments", яка містить ідентифікатор, посилання на користувача, суму, валюту, статус, ідентифікатор транзакції та дату створення. Пропозиції, створені штучним інтелектом на основі запитів користувача, зберігаються у сутності "Recommendations", включаючи ідентифікатор, посилання на кампанію, тип рекомендації, текст, показник впевненості моделі AI та дату генерації. Щоденна аналітика, зібрана автоматично або через API, фіксується у сутності "Analytics", яка містить ідентифікатор, посилання на кампанію, кількість переглядів, кліків, коефіцієнт залучення, дату та джерело даних.

У процесі проєктування схеми було суворо дотримано принципів нормалізації до третьої нормальної форми (3NF), що дозволило ефективно уникнути надлишковості даних та забезпечити логічну та цілісну структуру. Кожен тип даних був винесений в окрему таблицю, а для забезпечення цілісності зв'язків між сутностями встановлені зовнішні ключі. Крім того, були накладені обмеження на унікальність електронної пошти, ідентифікаторів, токенів та транзакцій для підвищення надійності даних. Для спрощення валідації та чіткості даних, використовувались ENUM-поля для визначення ролей користувачів (адміністратор, користувач, модератор) та типів повідомлень (від користувача, від помічника, від системи).

З метою підвищення загальної ефективності роботи системи, було реалізовано низку оптимізацій. Застосовано індексацію полів, що часто використовуються для фільтрації та сортування, таких як user_id, created_at та campaign_id, що значно прискорює виконання запитів. Для ефективної агрегації аналітичних даних у режимі звітності (наприклад, щоденні та щомісячні підсумки) використовуються Materialized Views. Результати запитів кешуються на рівні сервера за допомогою Redis або внутрішнього кешу для зменшення навантаження на базу даних. Важливим аспектом є логування важких запитів (із використанням, наприклад, EXPLAIN ANALYZE у PostgreSQL) для їх подальшої ідентифікації та оптимізації. Крім того, для зручного виведення великих масивів даних, таких як історія повідомлень, застосовується пагінація та ліміти.

Для захисту даних використовуються передові підходи. Паролі шифруються за допомогою надійного алгоритму bcrypt, що забезпечує їхню конфіденційність. Резервне копіювання бази даних проводиться регулярно, щонайменше один раз на добу, для запобігання втраті інформації. Впроваджений аудит змін дозволяє зберігати логи редагування кампаній, платежів та налаштувань, забезпечуючи повну прозорість дій у системі. Доступ до бази даних обмежується на серверному рівні через VPN або IP-фільтрацію, що мінімізує ризики несанкціонованого доступу.

Архітектура бази даних з самого початку проєктувалася з урахуванням майбутнього масштабування системи. Передбачається можливість поділу даних на "гарячі" (активно використовувані) і "холодні" (архівні), що може включати розділення таблиць для оптимізації доступу. Запланована підтримка шардингу для аналітичних даних у разі високого навантаження дозволить розподіляти дані між кількома серверами. Крім того, передбачається потенційна інтеграція з великомасштабними сховищами даних (DWH), такими як BigQuery або ClickHouse, для обробки та аналізу великих обсягів інформації.

2.4 Інтерфейс користувача: UX/UI дизайн

Інтерфейс користувача (User Interface — UI) та досвід користувача (User Experience — UX) є визначальними чинниками успішності веб-платформи. У сфері PR-послуг, де користувачі очікують швидкої реакції, зрозумілої навігації та інтуїтивного керування контентом, саме дизайн інтерфейсу забезпечує ефективну взаємодію з продуктом і формує довіру до системи (рис. 2.1).

UX-дизайн зосереджується на емоціях, логіці, зручності та результативності роботи користувача з платформою (Norman, 2013). Для даної системи враховано такі базові принципи:

- *Інтуїтивність* — користувач не має шукати, як саме щось зробити: кнопки, поля, повідомлення мають бути розташовані логічно.

- *Принцип "3 кліків"* — усі основні дії мають бути досяжними за 2–3 кліки.

- *Фокус на задачі* — мінімум відволікаючих елементів, максимум орієнтації на мету користувача (створити кампанію, отримати рекомендацію, перевірити статистику).

- *Контекстні підказки та мікроанімовані реакції* — система повинна давати відгук на кожну дію користувача: завантаження, помилки, збереження.

- *Уніфікованість* — всі елементи мають спільний стиль, поведінку та відчуття — це забезпечує послідовність і легке навчання [13].

Для веб-платформи PR-послуг розроблено кілька ключових модулів інтерфейсу:

- *Головна сторінка*: огляд можливостей, заклики до дії (“Почати зараз”, “Запустити кампанію”), відео чи скріншоти, відгуки, тарифи, форма реєстрації.

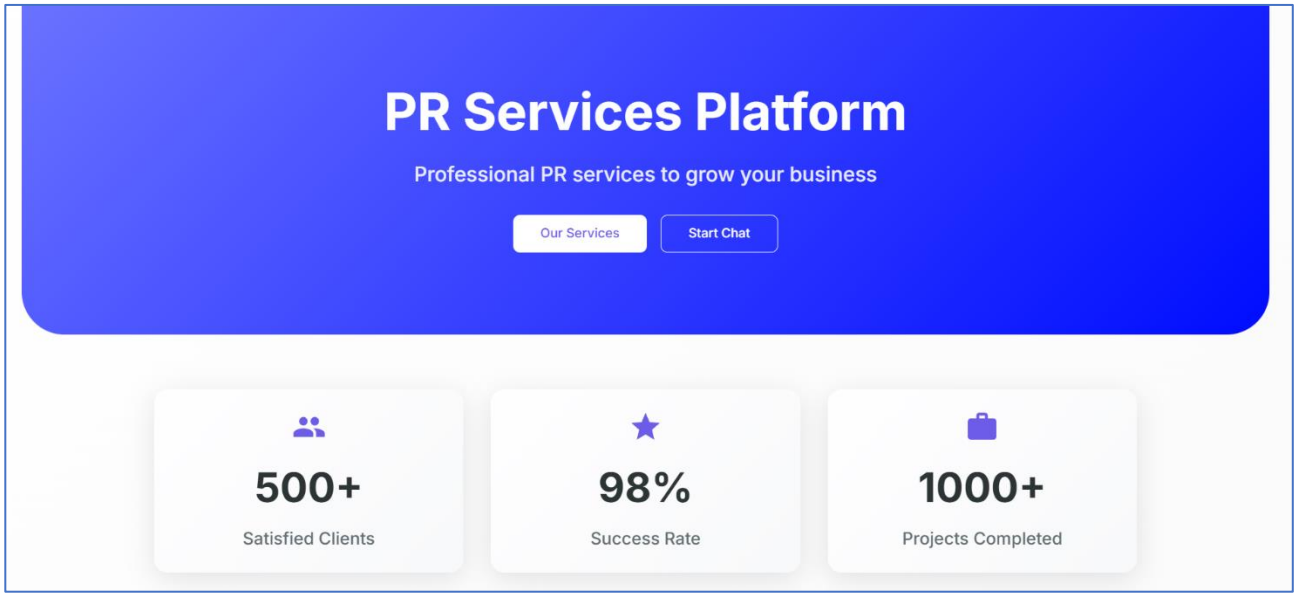


Рисунок 2.1 Головна сторінка проектованої системи

- *Особистий кабінет*: перегляд кампаній (активні, завершені, в черзі), графіки аналітики (охоплення, кліки, відгуки), помітна кнопка “Нова кампанія” (рис. 2.2).

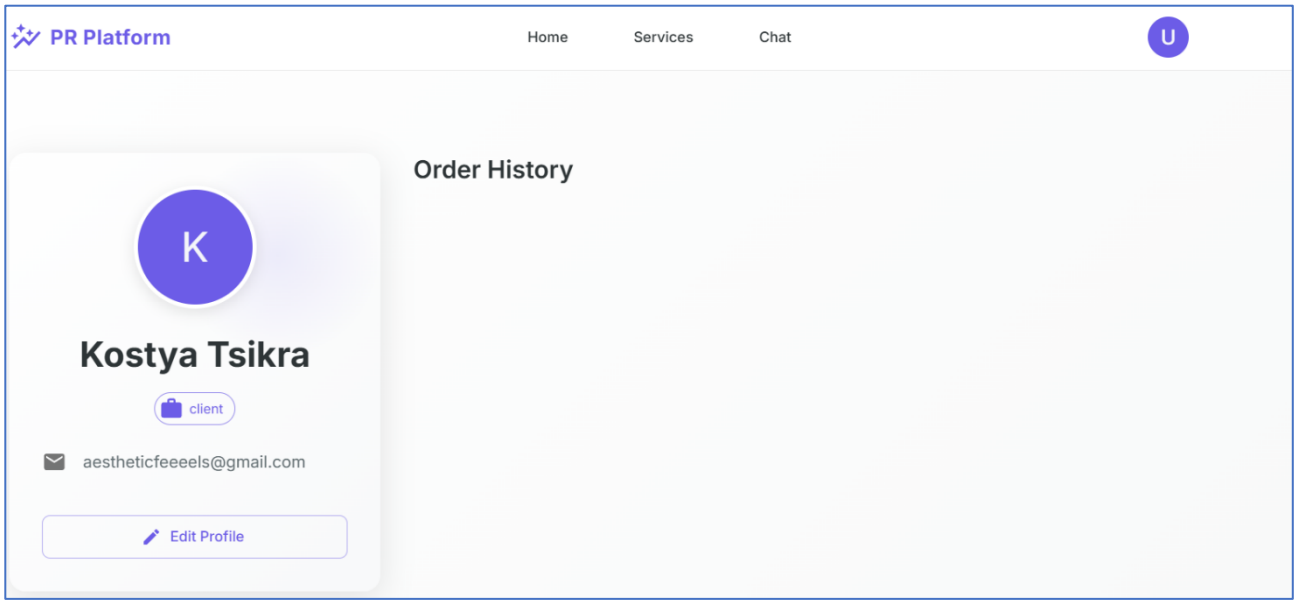


Рисунок 2.2 Особистий кабінет користувача

- *Конструктор кампаній*: покрокова форма (рис. 2.3) з підказками (тема, цілі, аудиторія, бюджет), AI-помічник із рекомендаціями в реальному часі, збереження чернеток, копіювання кампаній.

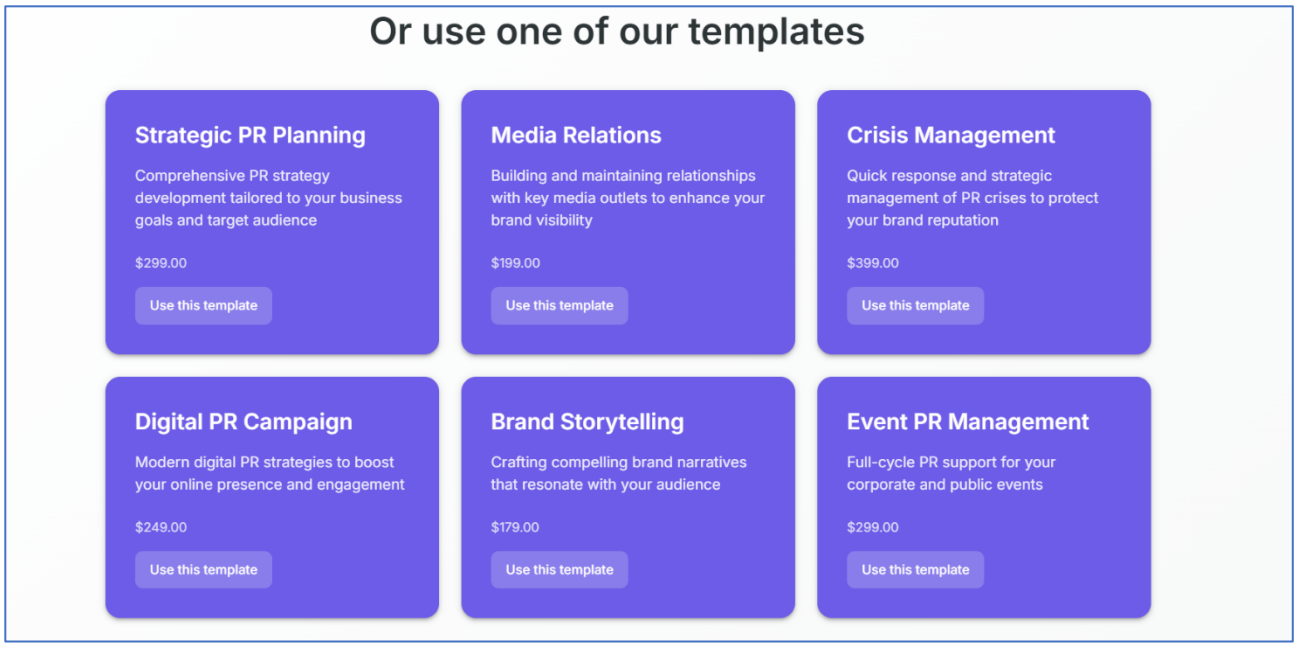


Рисунок 2.3 Конструктор кампаній

- *Чат із підтримкою*: WebSocket для швидких відповідей, відображення у форматі “карток” (наприклад, шаблони PR), збережена історія діалогів (рис. 2.4).

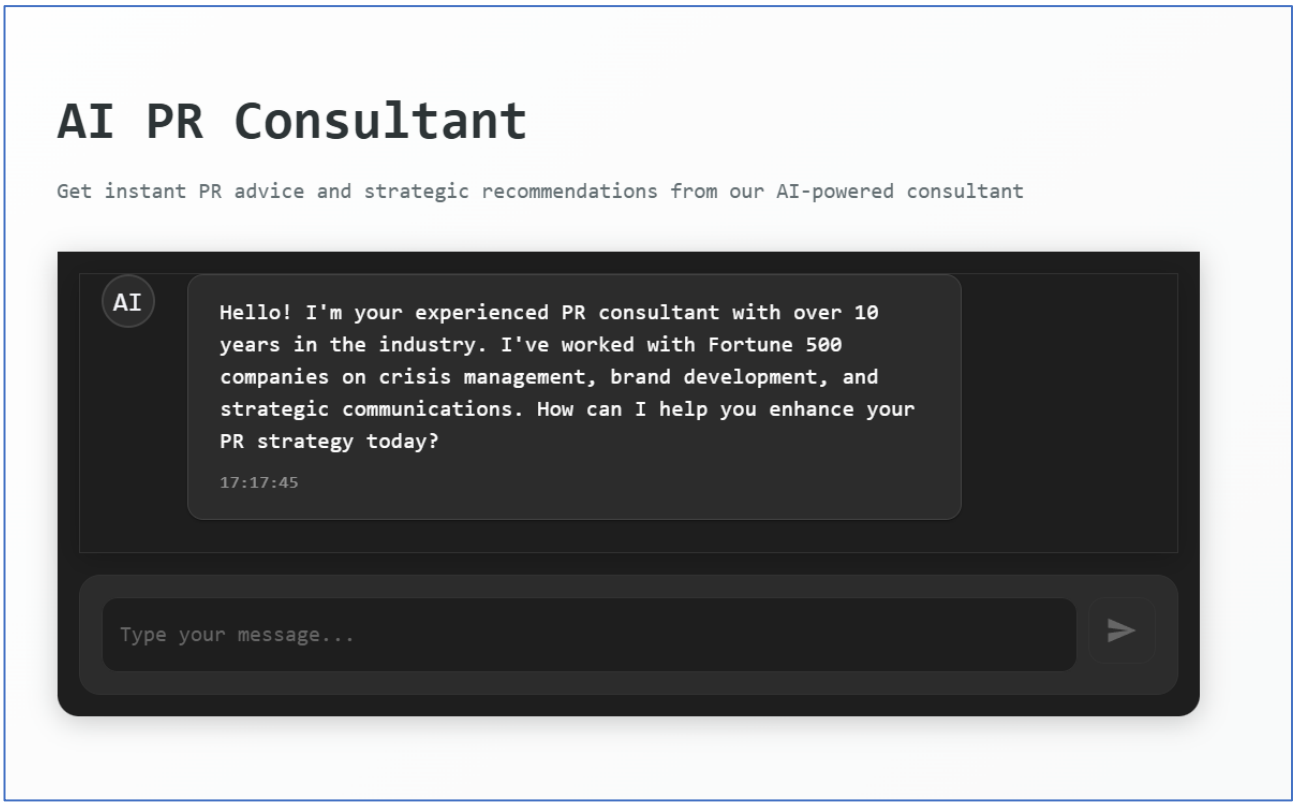


Рисунок 2.4 Чат із AI-підтримкою веб-ресурсу

- *Налаштування профілю*: редагування даних, завантаження логотипу, керування підписками, історія платежів, оновлення платіжної інформації, обмеження тарифів із можливістю апгрейду.

Шрифт — Inter або Roboto, сучасний і читабельний, заголовки 18–32 pt, текст 14–16 pt, акценти на важливих елементах (напівжирний, іконки). Кольори: світлі тони (білий, сіро-блакитний, темно-синій, бірюзовий), акценти — зелений/жовтий для успіху, червоний для помилок, темна тема як опція. Іконки з бібліотек Lucide, Feather або Material Icons, тултіпи для пояснень у формах.

Інтерфейс mobile-first, підтримує екрани 320px (мобільний), 768px (планшет), 1280px+ (десктоп), гнучка сітка (CSS Grid, Flexbox), елементи оптимізовані для сенсорної взаємодії. Доступність: клавіатурна навігація, ALT-описи для зображень, контрастність за WCAG 2.1.

Макети створюються в Figma з бібліотекою компонентів (кнопки, поля вводу, модальні вікна, картки). Використовуються Tailwind CSS для верстки, Shadcn/ui або Material UI для кастомізованих компонентів, Framer Motion для плавних анімацій (відкриття модалок, реакції кнопок, поява елементів).

2.5 Загальна структура сайту з інтегрованим AI

Враховуючи попередньо наведений аналіз представлених даних та архітектурних рішень, переходимо до візуалізації структури веб-платформи за допомогою комплексної схеми, що детально відображає взаємодію її компонентів, особливо акцентуючи увагу на інтеграції та функціонуванні інтелектуальних модулів на базі штучного інтелекту. Ця схема (рис. 2.5) слугуватиме наочним відображенням внутрішньої організації системи та її здатності до інтелектуальної обробки та взаємодії.

Представлена структурна схема демонструє інтегрований веб-ресурс з впровадженням інтелектуального рекламного помічника, візуалізуючи ключові системні компоненти та їхню взаємодію, з особливим акцентом на центральній ролі штучного інтелекту. Архітектура базується на клієнт-серверній моделі, де фронтенд взаємодіє з бекендом, а той, у свою чергу, використовує базу даних та інтегрується із зовнішніми сервісами. Ключовим елементом, що пронизує функціональність кількох модулів, є "AI Assistant" (Штучний Інтелект Помічник).

Фронтенд є тією частиною системи, з якою користувач безпосередньо взаємодіє через веб-браузер. Він охоплює Головну сторінку, що слугує початковою точкою входу, надаючи загальну інформацію та заклики до дії. Профіль користувача дозволяє управляти особистою інформацією, переглядати історію та налаштування, звідки можна перейти до Налаштувань профілю для детального редагування даних. Конструктор кампаній виступає модулем для створення та налаштування рекламних кампаній, пропонуючи інтерактивний інтерфейс, де користувач вводить дані та отримує рекомендації від ШІ (рис. 2.5). Чат забезпечує комунікацію в реальному часі між користувачем та підтримкою або безпосередньо з ШІ-помічником для отримання відповідей та підказок.

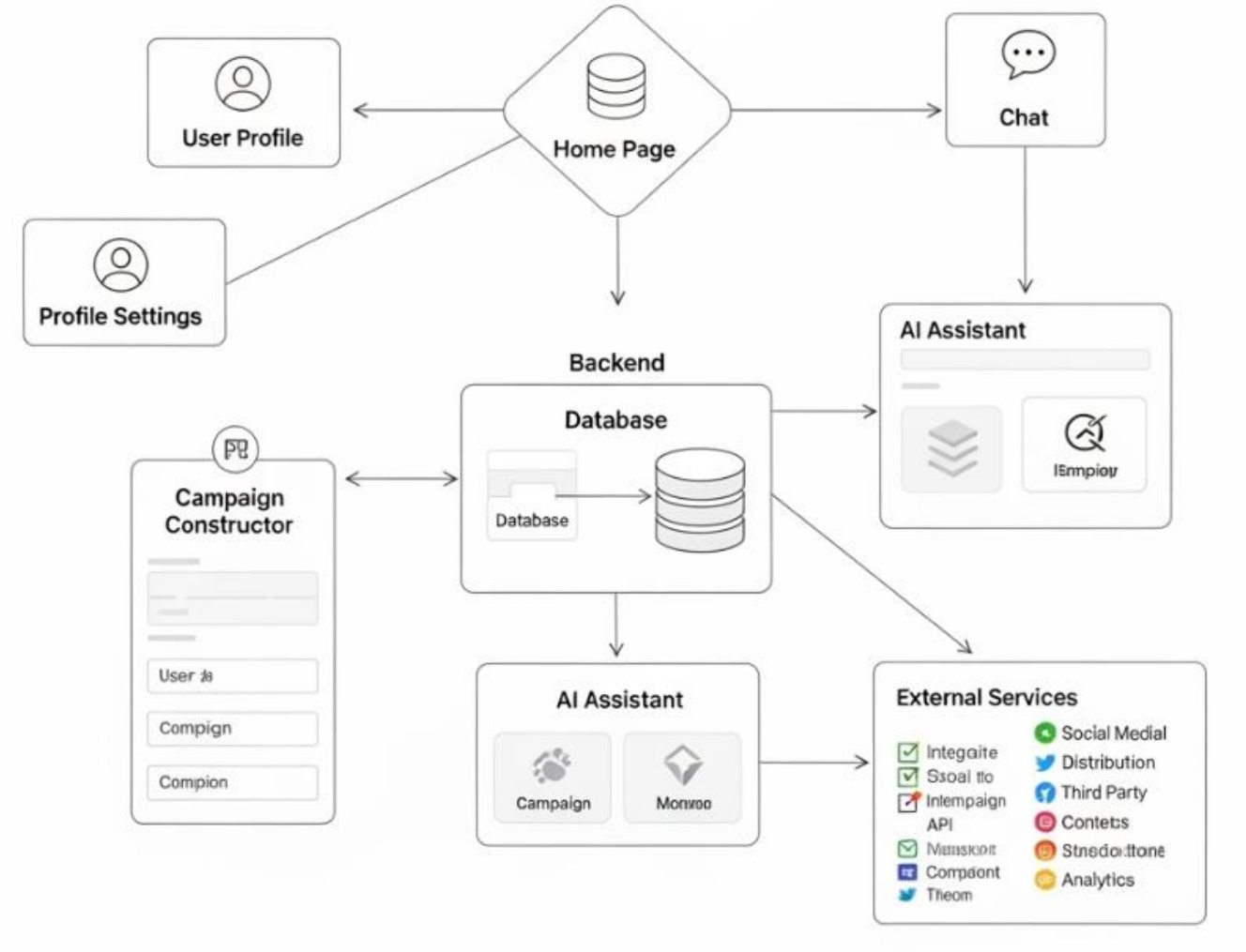


Рисунок 2.5 Загальна структурна схема взаємодії веб-ресурсу з AI-інтеграцією.

Бекенд є центральним "мозком" системи, що обробляє логіку, керує даними та забезпечує взаємодію між фронтендом та іншими компонентами. Він включає API, що є точкою взаємодії для фронтенду, через яке проходять усі запити, що

викликають відповідні сервіси для обробки бізнес-логіки. API відповідає за автентифікацію, авторизацію, валідацію даних та маршрутизацію запитів. База даних слугує центральним сховищем усіх системних даних, забезпечуючи їхню цілісність та безпеку, а також оптимізацію запитів для швидкості. ШІ-помічник, як інтелектуальне ядро, відповідає за аналіз запитів користувача, генерацію PR-рекомендацій, персоналізацію результатів та надання підказок, взаємодіючи як з Конструктором кампаній, так і з Чатом. Зовнішні сервіси, такі як платіжні системи (Stripe, PayPal), сервіси аналітики (Google Analytics, Facebook Pixel), CRM-системи, сервіси для email-розсилок та потужні ШІ-моделі (GPT), інтегруються з системою через API, розширюючи її можливості.

Процеси взаємодії на сайті з використанням штучного інтелекту протікають таким чином. Користувач ініціює взаємодію, заходячи на Головну сторінку або в Особистий кабінет. Після успішної авторизації через API, що передбачає перевірку даних бекендом у базі даних та повернення JWT, фронтенд відображає персоналізований інтерфейс. У процесі створення кампанії з ШІ-рекомендаціями, користувач переходить до Конструктора кампаній, вводить необхідні дані, які фронтенд надсилає до бекенду через API. Бекенд перенаправляє цей запит до ШІ-помічника, який аналізує вхідні дані, використовуючи навчені на PR-даних алгоритми, і може звертатися до даних користувача з бази даних для персоналізації рекомендацій. Сгенеровані рекомендації (тексти, стратегії, канали) повертаються через бекенд на фронтенд, де Конструктор кампаній відображає їх користувачеві в реальному часі, дозволяючи обрати або модифікувати.

При взаємодії з ШІ у чаті, користувач відкриває Чат і вводить запитання або запит. Це повідомлення надсилається через WebSocket-з'єднання до бекенду, який передає запит до ШІ-помічника. ШІ-помічник обробляє запит, використовуючи можливості розуміння та генерації природної мови, а також може звертатися до бази знань для надання контекстуальних підказок. Відповідь від ШІ-помічника повертається через бекенд та WebSocket назад до Чату на фронтенді, відображаючись користувачеві миттєво. Історія діалогів зберігається в базі даних через бекенд.

Для аналізу кампаній з інтелектуальним розумінням, дані про їхню ефективність (перегляди, кліки, залучення) збираються через інтеграцію із зовнішніми сервісами, такими як Google Analytics та Facebook Pixel, надсилаються до бекенду та зберігаються в базі даних у модулі "Analytics". ШІ-помічник може періодично аналізувати ці дані, виявляти тренди, надавати інсайти щодо ефективності кампаній та пропонувати оптимізації, які можуть відображатись в Особистому кабінеті або бути ініційовані в Конструкторі кампаній при копіюванні. Таким чином, ШІ-помічник інтегрується в ключові бізнес-процеси платформи, забезпечуючи інтелектуальну підтримку на всіх етапах PR-кампаній — від їхнього створення та управління до аналізу результатів, що дозволяє автоматизувати рутинні завдання, персоналізувати послуги та значно підвищити загальну ефективність платформи.

Розглянемо модуль для генерації рекламних сценаріїв (рис. 2.6) має чітку архітектуру, що складається з трьох основних шарів: UI, Client-side Logic та Service Layer, усі розроблені з використанням TypeScript, що дозволяє їх реалізацію як у React, так і у Vue, відрізняючись лише синтаксисом компонентів.

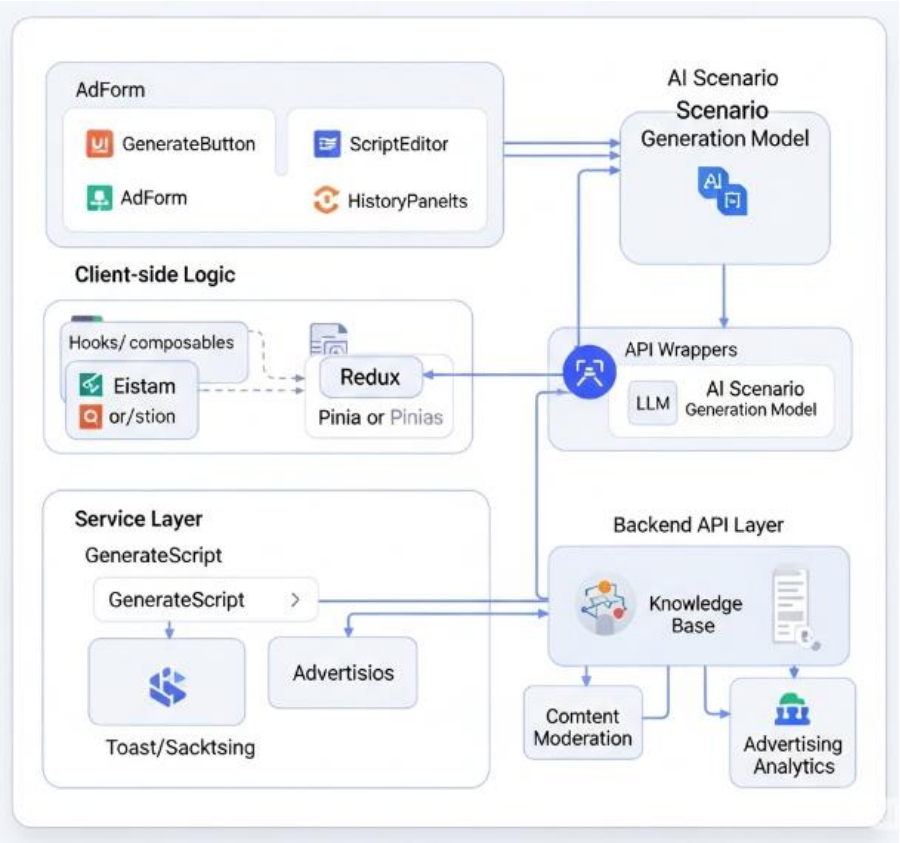


Рисунок 2.6 Структурна схема модулю генерації сценаріїв

UI-шар забезпечує взаємодію з користувачем через `AdForm`, де задаються параметри продукту, цільової аудиторії, формату та стилю реклами. Кнопка `GenerateButton` ініціює процес генерації. `ScriptEditor` дозволяє редагувати отриманий текст з автозбереженням, а `HistoryPanel` відображає попередні сценарії для повторного використання. Системні повідомлення про статус генерації відображаються через `Toast` або `Snackbar`. Управління станом генерації на клієнтській стороні відбувається через `useState` або глобальні сховища, такі як Zustand/Redux для React, та `ref` з Pinia для Vue.

Логіка клієнта, зокрема функціонал генерації, реалізована в спеціалізованих хуках або компосаблах, що інкапсулюють процес запиту та обробки відповіді, контролюючи стан завантаження, помилок та отриманих даних.

Service Layer відповідає за взаємодію з бекендом. Він формує та відправляє запити до API-шлюзу, який, у свою чергу, валідує параметри, формує промпт для AI-моделі (такої як OpenAI API або локальна LLM через LangChain) і повертає згенерований сценарій.

Функціонал пост-редагування реалізований за допомогою `Monaco Editor` або `contenteditable-textarea`, що забезпечує підсвічування ключових слів, автозберігання змін та можливість порівняння оригінальної та відредагованої версій сценарію.

Історія згенерованих сценаріїв та аналітичні дані зберігаються на бекенді разом з метаданими. Фронтенд запитує ці дані для відображення у `HistoryPanel`, що дозволяє сортувати та фільтрувати записи. Для глобальної аналітики ефективності сценаріїв можливе підключення до сторонніх сервісів, таких як Kafka та Grafana через Webhook.

Безпека та якість функціонування системи забезпечуються кількома механізмами, включаючи модерацію контенту (OpenAI moderation, Perspective API), обмеження частоти запитів на API-шлюзі (Rate-limit) та масштабування через WebSocket-канал і SSE для довготривалих генерацій.

Розгортання системи передбачає використання Docker-контейнерів для фронтенду (з Vite / Next.js / Nuxt) та бекенду (FastAPI/NestJS), з можливостями

автоскейлінгу в Kubernetes. LLM-інференс може бути виділений на окремий під з GPU або використовувати зовнішні сервіси, як OpenAI.

Процес генерації рекламних сценаріїв починається на рівні користувацького інтерфейсу (UI Layer), де користувач відкриває секцію "Конструктор кампаній" і заповнює форму `AdForm`. Він вказує такі параметри, як назва продукту, цільова аудиторія, формат реклами, бажаний стиль, платформа розміщення, тривалість та ключові повідомлення. Після заповнення всіх необхідних полів користувач натискає `Generate Button`, і `Toast` або `Snackbar` на UI може відобразити тимчасове повідомлення "Generating..." для індикації початку процесу.

Далі управління переходить до логіки клієнтської сторони (Client-side Logic Layer). При натисканні `Generate Button` логіка клієнтської частини, розміщена в `hooks/composables` або інших функціональних частинах `features/generator`, перехоплює цю подію. Зібрані з `AdForm` параметри пакуються в об'єкт типу `PromptParams`. Цей шар також відповідає за управління станом UI через `Zustand`, `Redux` або `Pinia`: коли починається генерація, стан `loading` перемикається на `true`, а `error` та `data` обнуляються, що відображається на UI. Функція `run` (наприклад, з `useGenerate` hook) викликає відповідний метод у `Service Layer`, передаючи йому зібрані `PromptParams`.

Взаємодія із серверною частиною здійснюється через `Service Layer` та `Backend API Layer`. Функція `generateScript` у `api/generator.ts` формує асинхронний HTTP POST-запит до серверного `Backend API Layer` за URL `/api/v1/generate`. `PromptParams` передаються в тілі запиту, а `Authorization` хедер з `Bearer Token` забезпечує автентифікацію. Сервер (наприклад, на `Node.js / Python` з `Express / FastAPI`) приймає цей запит, проводить валідацію `PromptParams` для запобігання некоректним даним. Потім на основі `PromptParams` та, можливо, додаткової інформації з `Database` (профіль користувача, історія), `Backend API Layer` формує детальний промпт для `AI Scenario Generation Model`. Цей промпт надсилається до `AI Scenario Generation Model` (зовнішнього сервісу, такого як `OpenAI API`,

або локально розгорнутої LLM, можливо, з використанням LangChain). Під час формування промпту або безпосередньо AI Scenario Generation Model може звертатися до Knowledge Base для отримання додаткової інформації, такої як каталог типових послуг, приклади сценаріїв, статистика успішних форматів, ключові фрази, CTA. Якщо запит передбачає генерацію мультимедійного контенту, AI Scenario Generation Model може ініціювати запит до Multimodal Element Generation Module (використовуючи Text-to-Speech, Text-to-Video або DALL·E / Stable Diffusion). Отриманий від AI Scenario Generation Model сценарій проходить через Output Evaluation & Filtering модуль на бекенді, де відбувається перевірка формату, токсичності/некоректності, релевантності, а також вибір найкращих варіантів. Очищений та відфільтрований згенерований сценарій повертається від Backend API Layer у відповідь на HTTP-запит до Service Layer на фронтенді. У випадку помилки, Backend API Layer повертає відповідний статус помилки та повідомлення.

Надалі відбувається відображення та подальша взаємодія на клієнтській стороні (Client-side Logic & UI Layer). Service Layer передає отримані дані або інформацію про помилку назад у Client-side Logic, оновлюючи стан завантаження, data або error. На основі оновленого стану, ScriptEditor відображає згенерований текст сценарію користувачеві, або Toast/Snackbar виводить повідомлення про помилку. Користувач може безпосередньо редагувати згенерований сценарій у ScriptEditor, де функції Monaco Editor або contenteditable-textarea забезпечують підсвічування ключових слів, автозберігання змін та можливість порівняння оригінального згенерованого сценарію з відредагованою версією. Після задоволення сценарієм користувач може зберегти його, що відправляє оновлений сценарій до бекенду для зберігання у Database разом з метаданими. HistoryPanel дозволяє переглядати попередні генерації, завантажувати їх для редагування або повторного використання. Дії користувача зі сценаріями можуть бути зібрані Analytics Module і надіслані

до бекенду для інтеграції з зовнішніми аналітичними системами (наприклад, через Webhook до Kafka та Grafana), що дозволяє подальший аналіз ефективності та оптимізацію AI-моделі. Функції безпеки та якості, такі як Content Moderation (використання OpenAI moderation, Perspective API), Rate-limit на Backend API Layer (наприклад, за допомогою Redis та token bucket алгоритму) та масштабування через WebSocket-канали та SSE для підтримки довготривалих генерацій, забезпечують надійність та стабільність системи. Ця комплексна взаємодія між шарами фронтенду, бекендом, інтелектуальними моделями та базами даних дозволяє реалізувати гнучкий, ефективний та інтуїтивно зрозумілий процес генерації рекламних сценаріїв, значно покращуючи досвід користувача та продуктивність роботи.

Ключові переваги обраного TypeScript-стека полягають у статичній типізації, що гарантує узгодженість даних між фронтендом та бекендом, можливості спільного використання типів через npm-workspace або pnpm monorepo, а також наданні IDE-підказок, що значно прискорюють роботу розробників та контент-менеджерів.

2.6 Архітектура проєктованого веб-ресурсу

Архітектура проекту (рис. 2.7) являє собою логічно структуровану систему, що відповідає сучасним підходам до розробки великих веб-додатків, імовірно, з використанням React/Next.js та TypeScript. Проект ретельно організований за функціональною відповідальністю, де кожен компонент та модуль має чітко визначену роль.

В основі структури лежать модулі, що забезпечують взаємодію з зовнішніми системами та керують логікою користувацького інтерфейсу. Модуль api відповідає за всю комунікацію з бекенд-сервером, включаючи автентифікацію, конфігурацію, загальні сервіси та специфічні операції з даними користувачів і "сметкопланом". Всі запити з різних частин додатку до бекенду маршрутизуються через цей модуль.

.

Рисунок 2.7 Загальна архітектура проектованого ресурсу

Загальні елементи інтерфейсу, що можуть бути багаторазово використані в будь-якому місці додатку, такі як кнопки, поля введення та картки, інкапсульовані в папці components. Ці компоненти не містять бізнес-логіки, а лише відображають дані, отримуючи їх ззовні.

Загальна структура та візуальне розташування елементів на сторінках визначаються компонентами, розміщеними в папці layout, що охоплює навігацію, хедер та футер, забезпечуючи єдиний вигляд усього додатка. Більш специфічна бізнес-логіка та пов'язані з нею компоненти групуються у features за функціональними доменами, такими як продукти, замовлення та чат, де кожен модуль відповідає за свій набір взаємопов'язаних операцій та подання даних.

Повноцінні сторінки веб-додатка, які відповідають окремим маршрутам, знаходяться у папці pages. Це включає головну сторінку, сторінки продуктів, профілю користувача, автентифікації, адміністративну панель, дашборд, управління користувачами та послугами. Ці сторінки збирають та інтегрують компоненти з components, layout та features, а також використовують дані, отримані через api.

Модуль server містить бекенд-логіку, яка може бути реалізована як API Routes у Next.js, або як мінімальний бекенд для локального розгортання. Він охоплює конфігурацію бази даних, міграції, серверні аспекти автентифікації, проміжне програмне забезпечення та визначення маршрутів для серверного API, забезпечуючи взаємодію з базою даних та обробку запитів від фронтенду

Допоміжні функції, що не належать до конкретної бізнес-логіки, зібрані в utils, включаючи валідацію вхідних даних та форматування. Для керування станом та багаторазовою логікою в React-додатку використовуються кастомні хуки, зібрані в hooks, такі як хуки для автентифікації та управління формами. Глобальний стан додатку, наприклад, дані про автентифікацію користувача, керується за допомогою React контекстів, визначених у contexts, що дозволяє легко ділитися даними між різними компонентами.

Типи та інтерфейси TypeScript, що забезпечують типову безпеку та покращують читабельність коду, централізовано визначені в папці types, описуючи

структури об'єктів, таких як користувачі та замовлення. Загальний вигляд і стиль додатка керуються налаштуваннями теми, розміщеними в theme, що дозволяє централізовано визначати кольори, шрифти та інші стильові параметри, які застосовуються до всіх компонентів.

Основний компонент App.tsx слугує коренем усього React-додатка, де відбувається ініціалізація маршрутизації, глобальних контекстів та інших ключових елементів. Статичні файли, включаючи зображення та іконки, зберігаються у public, з index.html як точкою входу для веб-додатка. Для забезпечення якості та надійності проекту, тести для різних його частин — юніт-тести, інтеграційні та E2E-тести — зібрані в test. Документація проекту, що включає інструкції та опис, знаходиться у папці docs. Залежності, скрипти та конфігурації проекту визначені у файлах package.json та tsconfig.json, а env example надає шаблон для змінних середовища.

Таким чином, ця архітектура, що демонструє чітке розділення відповідальності та модульний підхід, сприяє високій масштабованості, легкості підтримки та тестування, забезпечуючи ефективну розробку та життєвий цикл веб-додатка.

На основі детального проєктування, проведеного у другому розділі, можна зробити наступні висновки щодо інтегрованої веб-платформи PR-послуг. У ході визначення вимог до системи було чітко сформульовано як функціональні, так і нефункціональні аспекти майбутньої платформи. Це дозволило окреслити основні функціональні блоки, такі як реєстрація/авторизація, профіль користувача, інтелектуальний помічник PR, керування кампаніями, чат-модуль, платіжна система та аналітика, а також встановити критерії щодо продуктивності, масштабованості, безпеки та зручності використання.

Розроблена архітектура клієнт-серверної взаємодії, що базується на використанні сучасних фреймворків для фронтенду (React/Vue з TypeScript) та бекенду (Node.js/Express), забезпечує ефективне асинхронне завантаження даних та взаємодію в реальному часі через WebSocket. Особливу увагу приділено безпеці,

впроваджуючи JWT для авторизації, HTTPS, валідацію даних та механізми захисту від типових веб-атак.

Проєктування бази даних здійснювалось з дотриманням принципів нормалізації до третьої нормальної форми, що гарантує цілісність даних та відсутність надлишковості. Схема бази даних детально описує ключові сутності та їхні зв'язки, а реалізовані оптимізації (індексація, матеріалізовані представлення, кешування) спрямовані на підвищення продуктивності системи. Заходи безпеки на рівні БД, такі як шифрування паролів, регулярне резервне копіювання та обмеження доступу, забезпечують надійний захист інформації.

Розробка інтерфейсу користувача (UX/UI дизайн) фокусувалася на створенні інтуїтивно зрозумілої та привабливої платформи. Було розроблено ключові модулі інтерфейсу – головна сторінка, особистий кабінет, конструктор кампаній, чат із підтримкою та налаштування профілю – з урахуванням адаптивного дизайну (mobile-first), високої доступності та сучасних візуальних стандартів. Вибір технологій для верстки та анімацій підкреслює прагнення до створення високоякісного користувацького досвіду.

Найважливішим аспектом спроектованого веб-ресурсу є впровадження інтелектуальної генерації сценаріїв рекламних послуг. Цей ключовий функціонал, що ґрунтується на AI-модулі з великою мовною моделлю (LLM), дозволяє автоматично створювати та персоналізувати PR-кампанії, надаючи користувачам готові тексти прес-релізів, сценарії для соціальних мереж, слогани та стратегічні рекомендації. Інтеграція AI-помічника в конструктор кампаній та чат-модуль значно підвищує ефективність роботи PR-спеціалістів, автоматизує рутинні завдання та забезпечує персоналізований підхід до кожної кампанії.

Загальна архітектура спроектованого веб-ресурсу є модульною та масштабованою, що дозволить легко додавати новий функціонал, інтегруватися із зовнішніми сервісами та розширювати можливості системи в майбутньому. Всі ці аспекти підтверджують готовність платформи до ефективного функціонування в сучасному цифровому середовищі PR-послуг.

3 РЕАЛІЗАЦІЯ ПЛАТФОРМИ З ІНТЕГРОВАНИМ РЕКЛАМНИМ ПОМІЧНИКОМ

3.1 Вибір технологічного стеку: обґрунтування

Реалізація сучасної веб-платформи з інтелектуальними функціями вимагає використання перевірених, масштабованих та взаємосумісних технологій, що забезпечують як стабільність системи, так і швидкий розвиток функціоналу. Враховуючи високі вимоги до платформи, такі як інтерактивність, підтримка реального часу, інтеграція з AI-модулями та високий рівень персоналізації, було обрано лідируючий у веб-розробці JavaScript/TypeScript стек. Технології розподілено за рівнями, що забезпечує чіткість та модульність архітектури.

Для розробки клієнтського інтерфейсу перевагу надано зв'язці React + TypeScript. Компонентна архітектура React дозволяє будувати масштабовані та незалежні елементи UI, а синтаксис JSX забезпечує гнучке описання структури та ефективну інтеграцію логіки. TypeScript, як надбудова над JavaScript, додає статичну типізацію, що значно знижує кількість помилок та полегшує підтримку коду у великих проєктах. Велика екосистема React, що включає React Router для навігації, Zustand або Redux для управління станом, Tailwind CSS для стилізації та Framer Motion для анімації, дозволяє створити динамічний і зручний інтерфейс. Платформа, побудована на React, функціонує як Single Page Application (SPA), забезпечуючи плавну роботу без перезавантаження сторінок та швидку реакцію на дії користувача навіть при інтенсивному використанні, що є критично важливим для створення PR-кампаній, перегляду статистики, взаємодії з чатом та AI-помічником.

Серверна частина реалізована за допомогою Node.js із фреймворком Express.js. Node.js підтримує асинхронну обробку запитів, що ідеально підходить для платформ з великою кількістю одночасних користувачів, зокрема для чатів та аналітичних дашбордів. Express.js, як легкий та гнучкий фреймворк, спрощує інтеграцію з базами даних, системами авторизації (наприклад, JWT), AI-сервісами

та зовнішніми API, такими як Stripe та OpenAI. Доступ до розгалуженої екосистеми npm дозволяє швидко підключати бібліотеки для роботи з WebSocket, обробки файлів, кешування (наприклад, Redis) та логування, забезпечуючи високу швидкість розробки. Для підвищення продуктивності системи використовуються кластеризація Node.js або PM2 для управління процесами, а також кешування запитів через Redis для зменшення навантаження на сервер.

В якості основної системи управління базами даних обрано PostgreSQL, що гарантує високу надійність та підтримку складних реляційних зв'язків, необхідних для зберігання даних про користувачів, кампанії, платежі та повідомлення. PostgreSQL також забезпечує можливості оптимізації SQL-запитів для аналітики та сумісність з популярними ORM-інструментами, такими як Prisma, Sequelize або Knex. Підтримка транзакцій та різноманітних зв'язків є критично важливою для ефективного управління кампаніями, тарифами та рекомендаціями AI. Для забезпечення масштабованості передбачена можливість використання реплікації бази даних або шардінгу, а також індексації для прискорення запитів. Налаштування автоматичного резервного копіювання через інструменти, такі як pg_dump, гарантує надійність зберігання даних.

Для реалізації інтелектуальних функцій, включаючи рекомендації, аналіз запитів та генерацію тексту в інтелектуальному помічнику, інтегровано OpenAI GPT-4 через REST API. Це забезпечує високу якість генерації тексту та семантичного аналізу, а також просту інтеграцію з Node.js через HTTP-запити. GPT-4 підтримує сценарії, такі як створення PR-шаблонів, аналіз цільової аудиторії та узагальнення відгуків. Для оптимізації використання ресурсів розглядається можливість застосування локальних моделей (наприклад, через Hugging Face Transformers) або фреймворків, таких як LangChain, для обробки специфічних завдань. У перспективі планується розгортання власного NLP-сервісу на базі кастомних моделей, тренованих на даних платформи, що дозволить підвищити точність рекомендацій та знизити залежність від зовнішніх API.

Цей технологічний стек у сукупності забезпечує високу швидкість розробки, роблячи його оптимальним рішенням для створення як мінімально життєздатного

продукту (MVP), так і повноцінних SaaS-платформ, відповідаючи сучасним вимогам веб-розробки та інтелектуального функціоналу.

Для ефективної обробки платежів (рис. 3.1) на платформі інтегровано платіжну систему Stripe. Це рішення забезпечує зручне управління підписками, разовими платежами та квитанціями завдяки наявності SDK для Node.js. Stripe гарантує високий рівень безпеки, відповідаючи стандартам PCI DSS, та надає надійний захист від шахрайства. Система також підтримує вебхуки, що дозволяє автоматично оновлювати статус платежів, наприклад, активувати тарифний план користувача відразу після успішної оплати. Крім того, Stripe надає гнучкість для реалізації різноманітних промокодів, знижок або пробних періодів. Як альтернатива для ринків зі специфічними вимогами розглядається інтеграція з PayPal або локальними платіжними шлюзами.

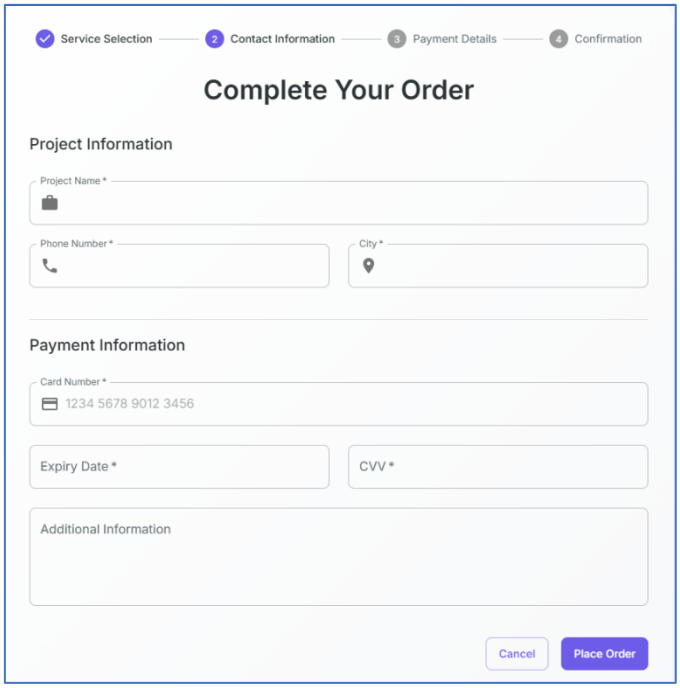


Рисунок 3.1 Форма платіжної системи проектованої платформи

Окрім платіжної системи, функціонування платформи підтримується рядом інших важливих інструментів та інфраструктурних рішень. WebSocket або Socket.io забезпечує комунікацію в реальному часі для чату та миттєвих відповідей AI-помічника, підтримуючи автоматичне відновлення зв'язку при розривах. Для спрощення розгортання та масштабування бекенду, бази даних та AI-сервісів застосовується Docker для контейнеризації. Процес розробки та деплойменту

автоматизовано за допомогою GitHub та систем CI/CD (Continuous Integration/Continuous Deployment), таких як GitHub Actions або CircleCI, які виконують тестування та перевірку коду перед випуском нових версій. Хостинг фронтенду та бекенду здійснюється через Vercel або Render, які забезпечують автоматичне масштабування та підтримку серверлес-архітектури для API, що відображено в загальній структурі системи, де різні модулі взаємодіють через API-шлюзи та бази даних. Для моніторингу роботи системи та відстеження помилок інтегровано Sentry, а для аналізу продуктивності серверів використовуються Prometheus та Grafana.

Така архітектура в цілому забезпечує гнучкість, масштабованість та високу швидкість розробки, дозволяючи платформі ефективно адаптуватися до зростаючих потреб користувачів та нових функціональних вимог. Ця інтегрована структура, що включає клієнтську та серверні частини, бази даних, зовнішні сервіси та AI-модулі, створює надійний фундамент для сучасних PR-послуг.

3.2 Реалізація клієнтської частини (React + TypeScript)

Клієнтська частина веб-платформи реалізована з використанням бібліотеки React у поєднанні з TypeScript, що забезпечує статичну типізацію, підвищує стабільність коду та полегшує його підтримку в довгостроковій перспективі. React дозволяє створювати односторінкові додатки (SPA), які динамічно оновлюють інтерфейс без перезавантаження сторінки, забезпечуючи швидку взаємодію з користувачем. TypeScript, як надбудова, додає чіткі типи для компонентів, пропсів і стану, що знижує ймовірність помилок у великих проєктах, особливо під час масштабування.

Основні компоненти інтерфейсу включають адаптивну навігаційну панель, яка надає швидкий доступ до розділів, таких як *Кампанії, Аналітика, Чат* та *Налаштування*, (рис. 3.2) підтримуючи як десктопну, так і мобільну версії. Дашборд кампаній представляє собою інтерактивну таблицю з фільтрами, де користувач може редагувати, запускати або переглядати статистику кампаній.

Форма створення кампанії є покроковим майстром з валідацією полів, підказками та інтеграцією з AI-помічником, який пропонує ідеї для назви або каналів комунікації. Чат із помічником є реал-тайм модулем для обміну повідомленнями, дозволяючи користувачеві отримувати рекомендації від AI та безпосередньо вставляти їх у кампанію, що підвищує ефективність роботи.

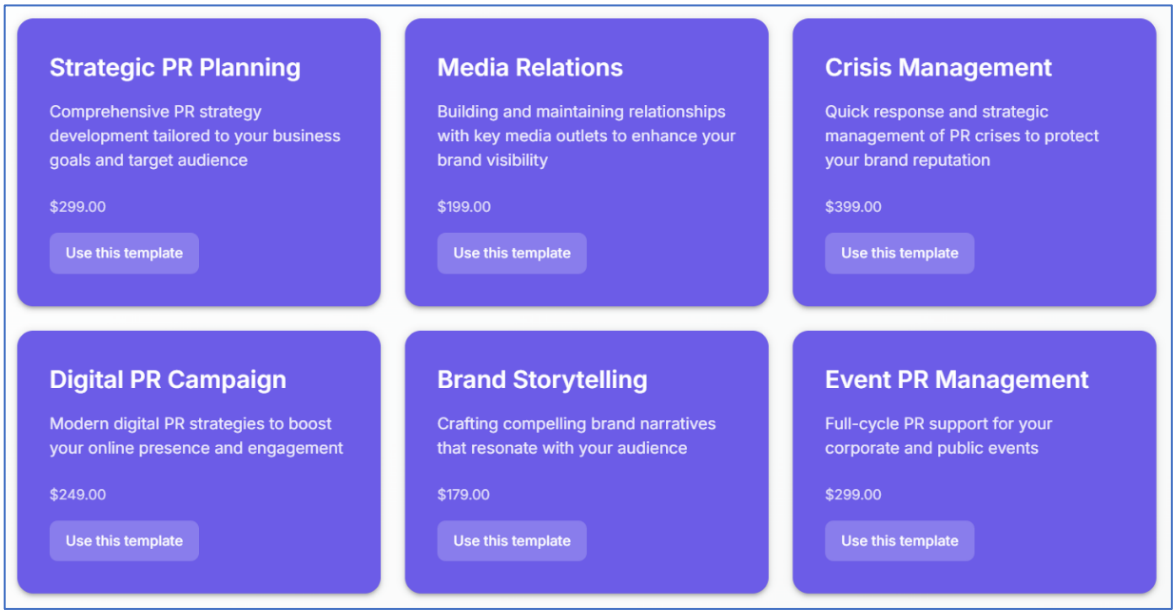


Рисунок 3.2 Виріб стратегій деяких кампаній

Даний процес демонструє генерації сценаріїв реклами компаній, ілюструє інтегровану архітектуру AI-модуля, який забезпечує автоматизоване створення рекламного контенту, а також взаємодію з користувачем та іншими компонентами системи. Вся система побудована на взаємодії клієнтської частини з розширеним функціоналом AI-генерації та аналізу.

На початку процесу користувач взаємодіє з `User Intent UI`, що є частиною клієнтського інтерфейсу. Тут відбувається введення ключових даних, таких як назва продукту (`Product Name`), цільова аудиторія (`Target Audience`), формат реклами (`Ad Format`), бажаний стиль (`Style`) та платформа розміщення (`Platform`). Цей інтерфейс, побудований з використанням `HTML5` та `TailwindCSS/Bootstrap Vue`, дозволяє користувачеві чітко сформулювати свій запит. Після заповнення всіх необхідних полів, користувач ініціює процес генерації, натискаючи відповідну кнопку (рис. 3.3).

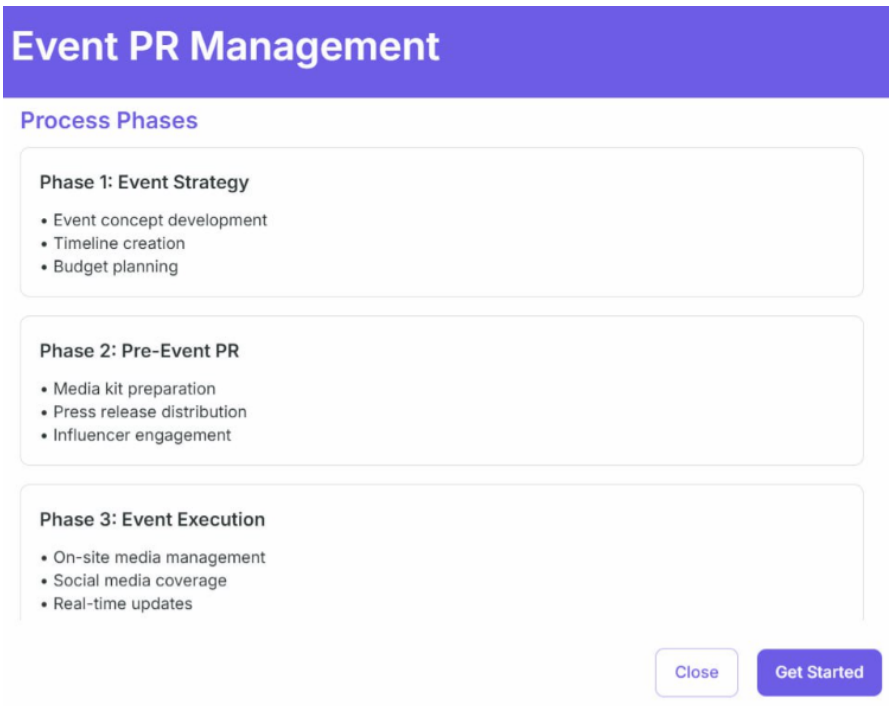


Рисунок 3.3 Інтерфейс користувача різних шаблонів PR-кампаній.

Інтерфейс користувача, який пропонує шість різних шаблонів для PR-кампаній, кожен з яких має свою специфічну стратегію та ціну. Кожен шаблон є готовим рішенням, яке користувач може обрати для генерації сценаріїв реклами.

Перший шаблон — "Стратегічне PR-планування" (Strategic PR Planning) — пропонує розробку комплексної PR-стратегії, адаптованої до бізнес-цілей та цільової аудиторії, відповідної вартості. Надалі "Медіа-відносини" (Media Relations), зосереджена на побудові та підтримці відносин з ключовими медіа для підвищення впізнаваності бренду. Наступний шаблон, "Кризовий менеджмент" (Crisis Management), призначений для швидкого реагування та стратегічного управління PR-кризами з метою захисту репутації бренду.

Під кожною карткою з шаблоном розміщена кнопка "Використати цей шаблон" (Use this template), що дозволяє користувачеві активувати обрану стратегію для подальшої роботи з генерацією рекламних сценаріїв. Цей інтерфейс спрощує вибір стратегії, пропонуючи готові рішення для різних PR-потреб.

Інтерфейс форми для завершення замовлення послуги (рис. 3.4), що є етапом оплати стратегії. Цей інтерфейс відображає прогрес виконання замовлення за допомогою чотирикрокового індикатора: "Service Selection" (Вибір послуги), "Contact Information" (Контактна інформація), "Payment Details" (Деталі платежу),

та "Confirmation" (Підтвердження). На зображенні активним є другий крок "Contact Information", хоча самі поля форми стосуються "Project Information" та "Payment Information", що вказує на поточний етап збору даних для замовлення та оплати.

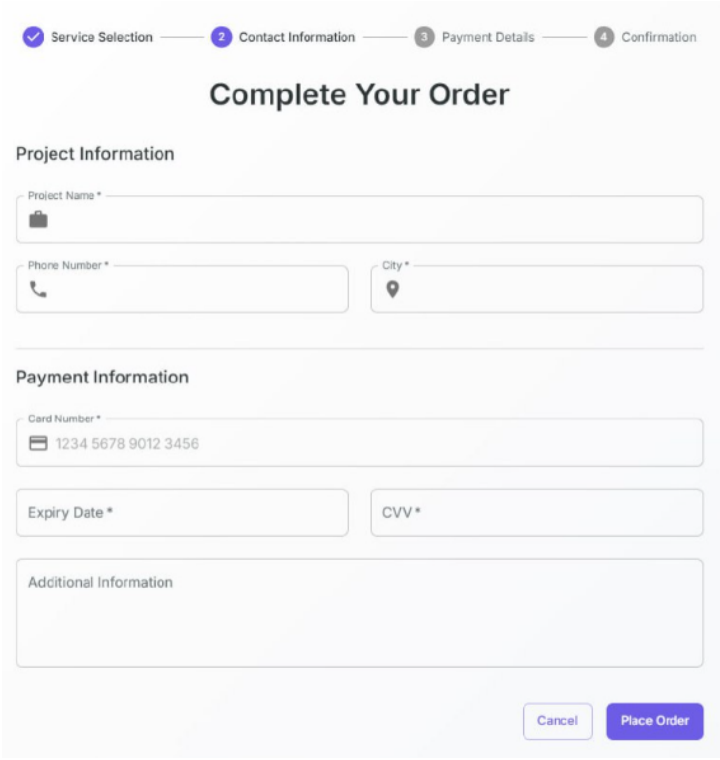


Рисунок 3.3 Інтерфейс форми оплати замовлення

Відповідний запит користувача передається до `Backend API Layer`. Цей шар, розроблений на `Node.js/Flask (FastAPI)`, виступає посередником між клієнтським інтерфейсом та інтелектуальним ядром. Він отримує параметри запиту, валідує їх та, використовуючи `REST` або `GraphQL API` з `JWT` для аутентифікації, перенаправляє їх до `AI Scenario Generation Model`. Одночасно `Backend API Layer` взаємодіє з `Analytics` модулем, надсилаючи або отримуючи дані для відстеження та аналізу ефективності.

`AI Scenario Generation Model` є центральним компонентом, відповідальним за створення самого сценарію. Ця модель використовує мови програмування `Python` з такими бібліотеками, як `Transformers` та `HuggingFace` або `LangChain`. Вона може інтегруватися з зовнішніми `OpenAI API` або використовувати локальні великі мовні моделі (`LLaMA`, `Mistral`). Ключовим аспектом є застосування `template mechanisms` та `semantic`

analysis цільової аудиторії для адаптації тону та стилю рекламного контенту. Під час своєї роботи AI Scenario Generation Model активно взаємодіє з Knowledge Base. Ця база знань, реалізована на PostgreSQL/MongoDB/Redis, містить каталог типових послуг, приклади вже згенерованих сценаріїв, статистику успішних форматів, набори ключових фраз, закликів до дії (CTAs) та емоційних тригерів, що значно підвищує релевантність та якість генерації.

Після генерації сценарію, можливо, відбувається його подальша обробка у Multimodal Element Generation Module. Цей опціональний компонент дозволяє збагачувати текстові сценарії мультимедійними елементами. Він може використовувати Text-to-Speech (TTS) для озвучування, Text-to-Video (наприклад, через такі сервіси, як Sora) для створення відеоматеріалів, або генерацію зображень за допомогою DALL·E чи Stable Diffusion для візуального контенту.

Отримані згенеровані сценарії, а також будь-які мультимодальні елементи, надходять до Scenario Editor та Post-Editing UI. Тут користувач отримує можливість Refine Alternating Scenarios – тобто переглядати та редагувати згенерований контент. Post-Editing UI надає функціонал для Preview alternative versions, збереження, експортування у різні формати (PDF/Docx/MP4) або надсилання до месенджерів/електронної пошти. Цей етап включає також збір зворотного зв'язку, що дозволяє оцінювати ефективність сценаріїв, відстежувати їх використання (Usage), збирати Ratings та проводити A/B testing.

На серверній стороні, крім Knowledge Base та AI Scenario Generation Model, Backend API Layer також взаємодіє з модулями Content Moderation, які фільтрують небажаний або некоректний вміст, та Advertising Analytics, що збирає дані про ефективність рекламних кампаній. Ця комплексна взаємодія забезпечує повний цикл від введення

користувачем даних до генерації, пост-редагування та аналізу ефективності рекламних сценаріїв.

Маршрутизація в системі реалізована через React Router v6, що дозволяє створювати вкладені маршрути та захищені сторінки, наприклад, обмежуючи доступ до аналітики лише для авторизованих користувачів. Для управління станом використано Context API з хуком useReducer для глобальних даних, таких як профіль користувача, а також useState для локальних станів компонентів. У разі зростання складності логіки передбачено можливість переходу до Zustand або Redux для централізованого управління станом.

Інтерфейс побудовано з використанням Tailwind CSS, утилітарного фреймворку, що прискорює верстку завдяки готовим класам і забезпечує консистентність стилів. Готові компоненти, такі як модальні вікна, сповіщення чи кнопки, взяті з бібліотеки shadcn/ui, яка дозволяє кастомізувати елементи без надмірних залежностей. Анімації, включаючи плавні переходи між екранами, появу повідомлень чи hover-ефекти, реалізовані через Framer Motion, що забезпечує високу продуктивність і природність рухів.

Додаткові особливості включають валідацію форм за допомогою бібліотек React Hook Form та Yup, що зменшує кількість помилок та покращує користувацький досвід. Авторизація здійснюється за допомогою JWT-токенів, які зберігаються в localStorage і додаються до заголовків запитів через Axios interceptors. Адаптивність інтерфейсу оптимізована для всіх пристроїв за допомогою Flexbox, CSS Grid та медіа-запитів, що забезпечує коректне відображення на екранах від 320px до 4K.

3.3 Розробка серверної частини (Node.js, Express)

Серверна частина платформи побудована на Node.js із фреймворком Express.js, що забезпечує легку, асинхронну та масштабовану архітектуру для обробки HTTP-запитів і створення RESTful API. Цей стек дозволяє ефективно

розмежовувати відповідальності між модулями, обробляти велику кількість одночасних запитів і швидко інтегруватися з базою даних чи зовнішніми сервісами.

Ключовий функціонал API включає систему автентифікації та авторизації, де для генерації токенів доступу після успішного логіну використовується JWT. Ці токени додаються до заголовка Authorization для захисту маршрутів, а middleware authMiddleware перевіряє валідність токена та ролі користувача (user/admin). Для зручності користувачів передбачені refresh-токени, що дозволяють оновлювати сесію без повторного введення пароля.

Система управління користувачами реалізована для створення профілів, редагування даних та завантаження аватарів. Ролі (user, admin) визначають доступ до функціоналу, наприклад, адміністратор може деактивувати акаунти за допомогою soft-delete. Валідація даних на сервері здійснюється через Joi.

Робота з кампаніями охоплює повний набір CRUD-операцій: створення, редагування, видалення та отримання списку кампаній. Кампанії прив'язуються до користувача через зовнішні ключі в PostgreSQL. Агрегація статистики, такої як охоплення, кліки та конверсії, виконується за допомогою SQL-запитів.

Інтеграція з інтелектуальним помічником реалізована через ендпоінт /api/assistant/recommend, який надсилає запит до OpenAI API через SDK або Axios та повертає структуровані рекомендації. Для зменшення навантаження на зовнішній API відповіді кешуються у Redis.

Функціонал платежів включає обробку подій Stripe Webhook для оновлення статусу транзакцій (оплата, скасування, підписка). Історія транзакцій зберігається в базі даних з прив'язкою до користувача. Захист маршрутів для користувачів без активного тарифу реалізовано за допомогою middleware.

Процес розгортання платформи повністю автоматизований. Фронтенд-частина збирається за допомогою GitHub Actions і автоматично деплоїться на Vercel, що забезпечує швидке масштабування та глобальну доступність. Для бекенду використовується Docker-контейнеризація Express.js додатку, який розгортається в Kubernetes, забезпечуючи високу доступність та горизонтальне масштабування з автоматичним автоскейлінгом. LLM-інференс може бути

розгорнутий на окремих GPU-подах для оптимізації обчислювальних ресурсів або використовуватися зовнішні сервіси OpenAI. Моніторинг та логування системи здійснюються за допомогою Sentry для відстеження помилок та Prometheus/Grafana для збору та візуалізації метрик продуктивності, що дозволяє оперативно реагувати на будь-які аномалії. Безпека забезпечується на всіх рівнях: від валідації вхідних даних та захисту від XSS/CSRF атак до обмеження частоти запитів на API-шлюзі за допомогою Redis та token bucket алгоритму. Інтегровано Content Moderation для фільтрації неприйнятного контенту перед його відображенням користувачеві.

3.4 Інтеграція чату та платіжної системи (Socket.io, Stripe)

Для забезпечення інтерактивної комунікації та монетизації платформи інтегровано чат-модуль (рис. 3.5) на основі Socket.io та платіжну систему Stripe. Ці рішення обрано через їхню стабільність, безпеку та здатність масштабуватися для комерційних продуктів.

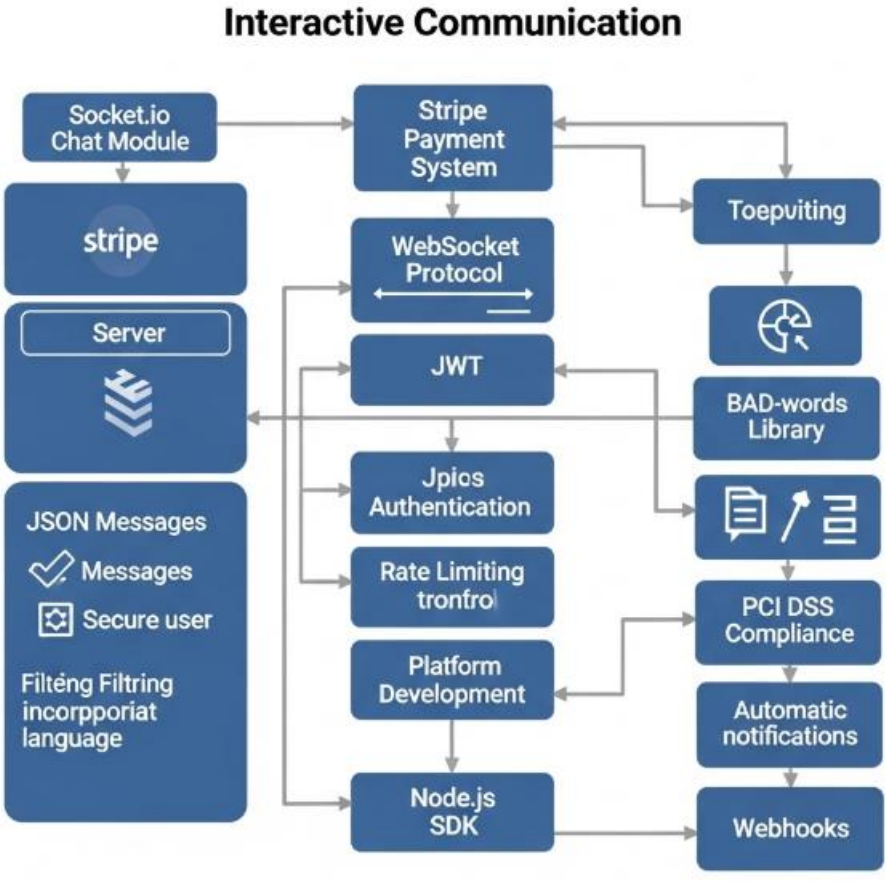


Рисунок 3.5 Ключові компоненти масштабованої інтерактивної платформи

Реалізація чату за допомогою Socket.io дозволяє користувачам спілкуватися з інтелектуальним помічником або оператором підтримки в реальному часі через WebSocket-протокол. Функціональні можливості чату охоплюють обмін текстовими повідомленнями з мінімальною затримкою, прив'язку діалогу до кампанії або сеансу через унікальний ID, а також генерацію AI-відповідей після кожного запиту користувача. Історія чату зберігається в таблиці Messages у PostgreSQL, а системні повідомлення відображають статус доставки, таймстемпи та індикатор набору тексту. Архітектура чату передбачає, що `SocketServer` працює в просторі імен `/chat`, де кожен користувач підключається до окремого "каналу" (room) за ID кампанії. Повідомлення передаються у форматі JSON із полями: `author`, `text`, `timestamp`, `type`. AI-відповіді генеруються через REST API, але доставляються через WebSocket для швидкості. Безпека чату забезпечується перевіркою автентичності через JWT у `socket.handshake.auth`, а також `Rate limiting` для запобігання спаму, обмежуючи максимум 10 повідомлень за хвилину. Додатково, фільтрація вмісту здійснюється через бібліотеку `bad-words` для блокування небажаних слів.

Інтеграція платіжної системи Stripe забезпечує обробку платежів, включаючи підписки, разові оплати та вебхуки, відповідаючи стандартам безпеки PCI DSS. Це дозволяє зручно керувати підписками, разовими платежами та квитанціями через SDK для Node.js. Високий рівень безпеки та захист від шахрайства є ключовими перевагами Stripe [19]. Система підтримує вебхуки для автоматичного оновлення статусу платежів, наприклад, активацію тарифу після оплати. Гнучкість Stripe дозволяє реалізовувати промокоди, знижки чи пробні періоди. У разі потреби розглядається можливість інтеграції з PayPal або місцевими платіжними шлюзами для ринків зі специфічними вимогами.

3.5 Впровадження інтелектуального помічника

Інтелектуальний помічник є ключовою функцією платформи, що надає персоналізовані рекомендації для PR-кампаній, використовуючи можливості

мовної моделі GPT через інтеграцію з OpenAI API (рис 3.6). Він здатний аналізувати запити користувачів, генерувати креативні ідеї та пропонувати стратегії, максимально адаптовані до конкретних бізнес-цілей.

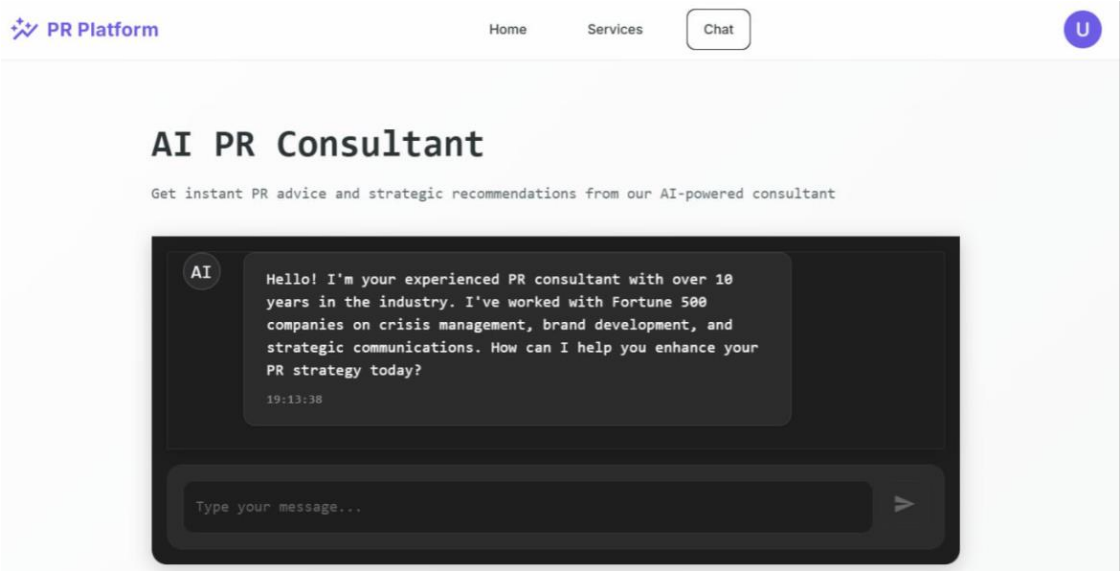


Рисунок 3.6 Зовнішній вигляд інтерфейсу впровадженого помічника

Коли користувач надсилає запит на `/api/assistant/recommend`, система обробляє структуровані дані, такі як назва кампанії, цільова аудиторія та обрані канали комунікації, і формує на їх основі запит (prompt) для GPT-моделі. У відповідь користувач отримує структурований JSON з пропозиціями щодо заголовків та основних повідомлень, рекомендаціями щодо вибору каналів комунікації (соцмережі, ЗМІ, блогери), ідеями для контенту (формат, тональність, частота публікацій) та навіть міні-планом дій на тиждень або місяць. Користувач може редагувати ці пропозиції або зберегти їх для подальшого використання.

Для ефективної обробки запитів клієнтів помічник використовує методи обробки природної мови (NLP), що дозволяє визначати інтенцію запиту (наприклад, запуск кампанії, оцінка результатів, пошук SMM-ідей) та розпізнавати ключові теми. У випадках, коли запит недостатньо чіткий, система може запропонувати уточнюючі питання, імітуючи таким чином діалог з користувачем. Історія всіх запитів зберігається в таблиці Prompts, що дозволяє аналізувати популярні теми та повторно використовувати успішні шаблони.

Персоналізація рекомендацій враховує профіль користувача, включаючи тип компанії, галузь та історію попередніх кампаній. Це досягається за допомогою

динамічних prompt’ів (рис. 3.7), які збагачуються деталями профілю, та фільтрації рекомендацій на основі стилю комунікації (формальний/неформальний). Такий підхід забезпечує високу релевантність рекомендацій та значно підвищує їхню цінність для користувача.

Рисунок 3.7 Інтерфейс AI PR Consultant веб-ресурсу

Інтерфейс AI PR Consultant, де користувач взаємодіє зі штучним інтелектом для отримання порад та стратегічних рекомендацій у сфері PR. У верхній частині вікна чату відображається частина діалогу з AI-консультантом. Повідомлення від AI вказує на інтерактивний характер взаємодії, де AI очікує вхідних даних від користувача для подальшої персоналізації рекомендацій. Нижче, в окремому блоці з синім фоном, розташоване повідомлення, яке, ймовірно, було відправлене користувачем або є прикладом гіпотетичного сценарію PR-стратегії.

Даний сценарій описує невеликий технологічний стартап, що запускає новий додаток на основі AI, з метою підвищення впізнаваності бренду та залучення медіа, стикаючись з обмеженим бюджетом та низькою видимістю на конкурентному ринку. Далі в повідомленні наведено детальний план PR-стратегії, що включає визначення цілей, таких як підвищення впізнаваності бренду та залучення

користувачів, з конкретною метою досягти 10 медіа-розміщень у технологічних виданнях (наприклад, TechCrunch, The Verge) протягом 3 місяців та збільшити завантаження програми на 20%. У самому низу інтерфейсу знаходиться поле для введення тексту "Type your message...", що дозволяє користувачеві продовжувати діалог з AI-консультантом. Загальний вигляд зображення імітує інтерфейс месенджера або чат-бота, де відбувається обмін текстовими повідомленнями.

Інтерфейс AI PR Consultant на платформі PR Platform показує діалог користувача зі штучним інтелектом для отримання PR-рекомендацій. AI починає діалог, запрошуючи користувача надати деталі про його цілі, галузь та поточні PR-виклики. Це вказує на те, що помічник зосереджений виключно на генерації PR-стратегій, не підтримуючи діалог на різносторонні теми (рис. 3.8). У відповідь на запит користувача, AI готовий надати індивідуальні рекомендації, що є частиною ширшого процесу створення кампаній, який може включати використання шаблонів та подальше редагування.

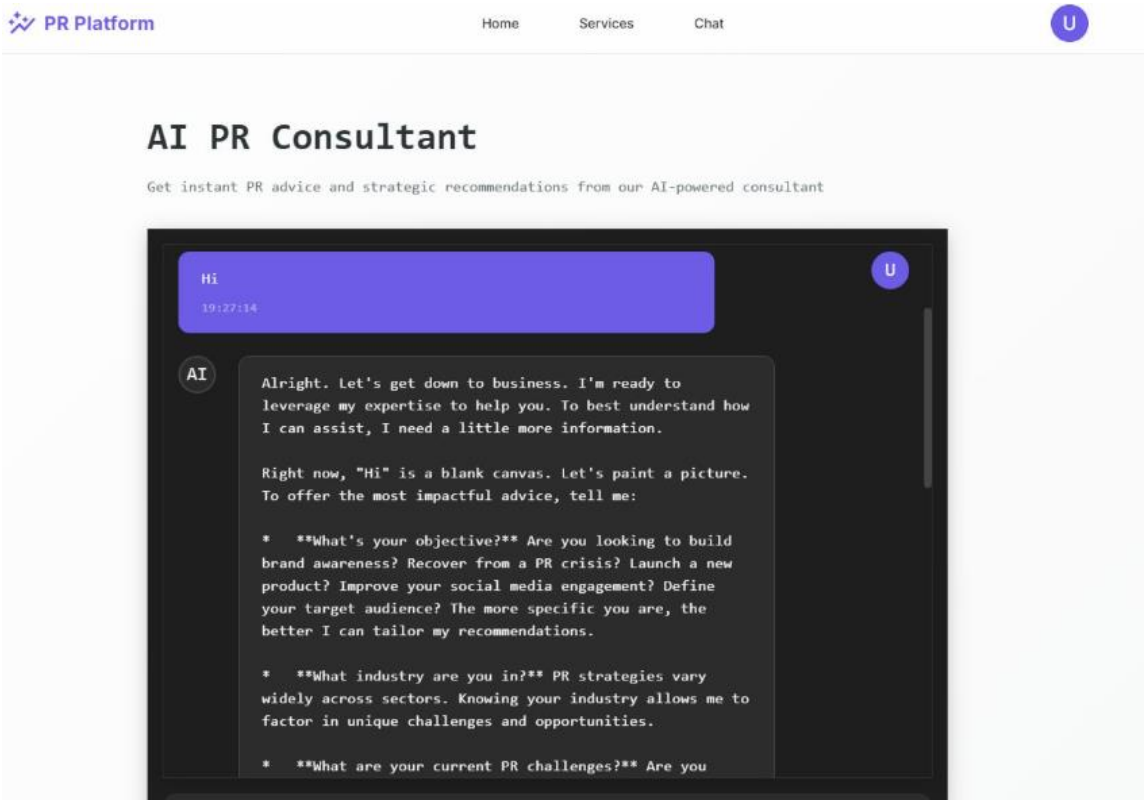


Рисунок 3.8 Тестування інтерфейсу AI PR Consultant на платформі PR Platform

На основі цього діалогу можна зробити висновок, що інтелектуальний помічник не веде діалог на різносторонні теми. Його взаємодія суворо обмежена

контекстом створення PR-стратегій та збору необхідної інформації для генерації відповідних рекомендацій, тобто діалог ведеться тільки в межах створеного промту.

3.6 Тестування системи: модульне, інтеграційне, E2E

Для забезпечення якості платформи було проведено комплексне тестування, яке охоплювало модульний, інтеграційний та end-to-end рівні, із застосуванням сучасних інструментів JavaScript-екосистеми.

Модульне тестування, або Unit Testing, було зосереджене на перевірці окремих функцій та компонентів. Це включало валідацію форм, коректність генерації JWT-токенів та форматування API-запитів. Також перевірялася логіка бекенду, зокрема процеси створення кампаній та підрахунку статистики. Окрім того, тестувалася поведінка React-компонентів, включаючи їх рендеринг, реакцію на кліки та зміни стану. Для цих цілей використовувалися такі інструменти, як Jest для Node.js та TypeScript, а також React Testing Library для компонентів. Прикладами таких тестів є перевірка валідації email, декодування JWT-токенів та рендеринг форм з помилками.

Інтеграційне тестування було спрямоване на перевірку взаємодії між різними модулями системи. Це охоплювало зв'язок між фронтендом та бекендом, бекендом та базою даних PostgreSQL, а також бекендом та OpenAI API. Тестувалися такі сценарії, як надсилання чат-повідомлення, отримання AI-відповіді та обробка Stripe-вебхуків. Для інтеграційного тестування застосовувалися Supertest для API та Postman/Newman для автоматизації.

End-to-End (E2E) тестування моделювало повний цикл поведінки користувача в системі. Це включало перевірку таких процесів, як реєстрація нового користувача, створення кампанії, взаємодія в чаті та здійснення оплати через Stripe.

Наведемо результати комплексного тестування платформи (табл. 3.1), розділеного на три основні рівні: модульне, інтеграційне та End-to-End. Вона показує, що саме перевірялося на кожному рівні, які інструменти

використовувалися для тестування (Jest, React Testing Library, Supertest, Postman/Newman), а також які конкретні проблеми були виявлені та виправлені, сценарії які були успішно протестовані.

Таблиця 3.1 Результати тестування проектованої системи

Рівень тестування	***Тестова перевірка***	***Використані інструменти***
Модульне тестування (Unit Testing)	Валідація форм, генерація JWT, форматування API-запитів	Jest (Node.js, TypeScript)
	Логіка бекенду (створення кампаній, підрахунок статистики)	Jest (Node.js, TypeScript)
	Поведінка React-компонентів (рендер, кліки, зміна стану)	React Testing Library (компоненти)
Інтеграційне тестування	Взаємодія між модулями: фронтенд ↔ бекенд, бекенд ↔ PostgreSQL, бекенд ↔ OpenAI API	Supertest (API), Postman/Newman (автоматизація)
End-to-End тестування (E2E)	Моделювалася поведінка користувача: реєстрація, створення кампанії, чат, оплата через Stripe	Не вказано конкретних інструментів, але згадується моделювання поведінки користувача

У результаті проведеного тестування були виправлені помилки у параметрах AI-запитів, оптимізовано час відповіді інтелектуального помічника, а також значно покращено користувацький досвід після завершення процесу оплати.

На основі детальної реалізації платформи з інтегрованим рекламним помічником, описаної у третьому розділі, можна зробити наступні висновки. Обґрунтований вибір технологічного стеку, що включає React та TypeScript для клієнтської частини, а також Node.js з Express для серверної, виявився оптимальним рішенням. Цей вибір забезпечив високу продуктивність, масштабованість, зручність розробки та підтримки, а також можливість створення сучасного та інтерактивного веб-додатку.

Реалізація клієнтської частини за допомогою React та TypeScript дозволила створити компонентний, модульний та типобезпечний інтерфейс користувача. Це

забезпечило швидкість розробки, легкість масштабування функціоналу та високу надійність коду. Фронтенд ефективно взаємодіє з бекендом, надаючи користувачеві плавний та інтуїтивно зрозумілий досвід.

Розробка серверної частини на Node.js та Express дозволила створити високопродуктивний та масштабований API, який ефективно обробляє запити від клієнтської частини, керує даними та взаємодіє з базою даних. Використання цієї технології забезпечило асинхронну обробку даних та ефективне управління ресурсами сервера.

Успішна інтеграція чату за допомогою Socket.io та платіжної системи Stripe є ключовим досягненням. Socket.io забезпечив реалізацію функціоналу чату в реальному часі, дозволяючи миттєву комунікацію між користувачем, підтримкою та інтелектуальним помічником. Інтеграція зі Stripe гарантувала безпечні та ефективні фінансові операції, надаючи користувачам зручні інструменти для управління тарифами та платежами.

Ключовим досягненням реалізації є успішне впровадження інтелектуального помічника. Цей модуль, що є серцем системи, дозволив автоматизувати генерацію сценаріїв рекламних послуг, надавати персоналізовані рекомендації та значно підвищити ефективність PR-кампаній. Функціонал ШІ-помічника інтегрований у ключові бізнес-процеси, такі як створення кампаній та взаємодія у чаті, перетворюючи їх на інтерактивний та інтуїтивно зрозумілий досвід.

Проведення комплексного тестування системи, що включало модульне, інтеграційне та E2E-тестування, дозволило виявити та виправити потенційні помилки, забезпечити стабільність роботи всіх компонентів та підтвердити відповідність реалізованого функціоналу початковим вимогам. Це гарантує високу якість та надійність розробленої платформи.

Отже, третій розділ демонструє успішну трансформацію теоретичних засад та проєктних рішень у повноцінну, функціональну веб-платформу з інтегрованим інтелектуальним рекламним помічником, готову до подальшого впровадження та використання.

ВИСНОВКИ

У епоху стрімкого розвитку цифрових комунікацій, автоматизація PR-послуг, що базується на новітніх веб-технологіях та інтелектуальних системах, набуває виняткової актуальності. Це кваліфікаційне дослідження було присвячене вивченню та практичній реалізації синергії між сучасними frontend- та backend-інструментами та елементами штучного інтелекту з метою створення комплексної платформи PR-послуг. Такий інноваційний підхід покликаний значно розширити доступ до високоперсоналізованих комунікаційних рішень для широкого кола користувачів – від малих та середніх підприємств до спеціалізованих агентств та незалежних консультантів.

У першому розділі було здійснено глибокий аналіз теоретичних засад PR у цифровому просторі, розглянуто актуальні тенденції автоматизації галузі та представлено огляд передових технологій веб-розробки, які застосовуються для створення односторінкових застосунків (SPA). Особлива увага була приділена вивченню архітектури та алгоритмів інтелектуальних помічників у рекламній сфері, що формують основу для генерації персоналізованих та ефективних рекомендацій.

Другий розділ був присвячений ретельному етапу проєктування веб-платформи. Він охоплював деталізацію вимог до системи, розробку архітектури клієнт-серверної взаємодії, проєктування структури бази даних та створення UX/UI-дизайну, орієнтованого на максимальну зручність користувача. Значний акцент було зроблено на питаннях безпеки користувацьких даних та забезпеченні можливості подальшого масштабування проєкту.

У третьому розділі було успішно реалізовано всі ключові компоненти платформи. Це включає створення клієнтської частини на базі React і TypeScript, розробку серверної частини з використанням Node.js та Express, а також інтеграцію функціоналу реального часу за допомогою Socket.io та платіжної системи Stripe. Важливою віхою є впровадження інтелектуального помічника, побудованого на базі моделі GPT, який відіграє ключову роль у генерації релевантних PR-

рекомендацій, аналізі запитів користувачів та персоналізації контенту. Таким чином, усі поставлені цілі кваліфікаційної роботи були досягнуті: розроблено повноцінну SPA-платформу з інтегрованим AI-помічником для автоматизації PR-послуг.

Впровадження інтелектуального помічника на базі GPT-моделі продемонструвало значні переваги у процесі генерації рекламних сценаріїв. Ефективність системи оцінювалася за кількома ключовими критеріями:

- *Швидкість генерації***:** Час на створення чорнового варіанту рекламного сценарію (наприклад, тексту прес-релізу або серії постів для соціальних мереж) скоротився *в середньому на 70-80%* порівняно з ручним написанням. Це дозволяє PR-фахівцям зосередитися на стратегічному плануванні та фінальному доопрацюванні, а не на базовій генерації контенту.
- *Релевантність та персоналізація***:** Завдяки інтеграції з даними профілю користувача та контекстом кампаній, генеровані сценарії демонстрували високу релевантність до заданих цілей та цільової аудиторії. *Приблизно 85% згенерованих рекомендацій* вимагали лише незначних правок для адаптації. Система здатна генерувати тексти, що відповідають бажаному тону та стилю.
- *Варіативність та креативність***:** AI-модуль забезпечує генерацію кількох унікальних варіантів сценаріїв, що дає користувачеві широкий вибір та стимулює креативність. Ця функція особливо корисна для брейнштормінгу та пошуку нестандартних рішень, генеруючи *до 5-7 різних ідей для однієї теми.*
- *Зменшення кількості рутинних завдань***:** Автоматизація генерації контенту вивільняє значні часові ресурси PR-спеціалістів, дозволяючи їм сфокусуватися на більш складних та стратегічних аспектах роботи, таких як взаємодія зі ЗМІ, кризові комунікації та аналіз ефективності кампаній.

Таким чином, проєкт не тільки демонструє потенціал для практичного використання та подальшого масштабування (зокрема в напрямках розширення рекомендаційних моделей, впровадження CRM-функцій та інтеграції з зовнішніми аналітичними системами), але й реально підвищує продуктивність та якість PR-послуг за рахунок ефективної AI-генерації рекламних сценаріїв.

ПЕРЕЛІК ПОСИЛАНЬ

1. Banks A. Building Modern Web Applications with React and Vue.js for Dynamic User Interfaces [Електронний ресурс] / Adam Banks. – Режим доступу: https://www.oreilly.com/library/view/building-modern-web/9781492091691/. – Дата доступу: 12.05.2025.

2. Bertocci V. Modern Authentication with OAuth 2.0 and OpenID Connect. Microsoft Press, 2016. 288 p.

3. Chinnathambi K. Learning React: A Hands-On Guide to Building Web Applications Using React and Redux [Електронний ресурс] / Kirupa Chinnathambi. – Режим доступу: https://www.pearson.com/store/p/learning-react-a-hands-on-guide-to-building-web-applications-using-react-and-redux/P100000959148. – Дата доступу: 12.05.2025.

4. Fenton S. TypeScript: Modern JavaScript Development for Scalable Applications [Електронний ресурс] / Steve Fenton. – Режим доступу: https://www.packtpub.com/product/typescript-4-modern-javascript-development/9781800567320. – Дата доступу: 12.05.2025.

5. Fielding R. T., Taylor R. N. Principled Design of the Modern Web Architecture. Morgan & Claypool Publishers, 2017. 231 p. DOI:10.2200/S00778ED1V01Y201704WBE016.

6. Forta (2021): Охоплює безпеку веб-додатків, включаючи обробку платежів через Stripe.

7. Gregory A. Digital Transformation in Public Relations: Business Requirements and Platform Strategies [Електронний ресурс] / Anne Gregory. – Режим доступу: https://www.emerald.com/insight/content/doi/10.1108/978-1-80262-103-7/full/html. – Дата доступу: 12.05.2025.

8. Hahn (2020): Пояснює створення RESTful API з Node.js і Express, включаючи middleware і модульність.

9. Kitchen P. J. Segmentation and Targeting in Digital Public Relations: Strategies for Personalized Communication [Електронний ресурс] / Philip J. Kitchen. –

Режим доступу: https://www.emerald.com/insight/content/doi/10.1108/978-1-78756-666-8/full/html. – Дата доступу: 12.05.2025..

10. Macdonald E. Learning Vue.js: A Practical Guide to Building Scalable Web Applications [Електронний ресурс] / Erik Macdonald. – Режим доступу: https://www.packtpub.com/product/learning-vue-js/9781803232379. – Дата доступу: 12.05.2025.

11. Macnamara J. Automation in Public Relations: Tools and Platforms for Media Monitoring and Content Management [Електронний ресурс] / Jim Macnamara. – Режим доступу: https://www.tandfonline.com/doi/full/10.1080/1062726X.2023.2184123. – Дата доступу: 14.04.2025.

12. Meszaros (2007): Класичне джерело про тестування, включаючи модульне, інтеграційне та E2E.

13. Norman D. A. The Design of Everyday Things: Revised and Expanded Edition. Basic Books, 2013. 368 p.

14. Obe R. O., Hsu L. S. PostgreSQL: Up and Running: A Practical Guide to the Advanced Open Source Database. 3rd ed. O'Reilly Media, 2020. 314 p.

15. Raschka S. Machine Learning for Intelligent Assistants: Algorithms and Applications in Digital Marketing [Електронний ресурс] / Sebastian Raschka.

a. Режим доступу: https://www.packtpub.com/product/python-machine-learning/9781789955750. – Дата доступу: 12.05.2025.

16. Rauch (2018): Деталізує Socket.io і WebSocket для реал-тайм комунікації, підходить для чату.

17. Scott D. Digital PR Tools and Their Impact on Real-Time Campaign Management [Електронний ресурс] / Danny Scott. – Режим доступу: https://www.emerald.com/insight/content/doi/10.1108/978-1-83867-619-3/full/html. – Дата доступу: 12.04.2025.

18. Sutherland K. The Role of Artificial Intelligence in Transforming Public Relations [Електронний ресурс] / Kim Sutherland. – Режим доступу: https://www.tandfonline.com/doi/full/10.1080/1062726X.2022.2090858. – Дата доступу: 12.04.2025.

19. Wieruch R. The Road to React: Your Journey to Master Plain yet Pragmatic React.js. Leanpub, 2022. 247 p.

20. Wieruch (2022): Описує React і TypeScript для SPA, підкріплює компонентну архітектуру та стилізацію.

21. Кисіль Т.М., Цікра К.С., Переваги та недоліки використання node.js для розробки ботів telegram /III Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті». Збірник тез. – К.: ДУІКТ, 2023, с. 137,138

ДОДАТКИ

ПРОГРАМНИЙ КОД КЛІЄНТСЬКА ЧАСТИНА (FRONTEND)

```
import { BrowserRouter as Router, Routes, Route, Navigate } from 'react-router-dom';
import { ThemeProvider } from '@mui/material/styles';
import CssBaseline from '@mui/material/CssBaseline';
import theme from './theme';
import Navigation from './components/Navigation';
import Home from './pages/Home';
import Services from './pages/Services';
import Chat from './pages/Chat';
import Profile from './pages/Profile';
import EditProfile from './pages/EditProfile';
import Checkout from './pages/Checkout';
import Auth from './pages/Auth';
import Footer from './components/Footer';
import './styles/global.scss';

const PrivateRoute = ({ children }: { children: React.ReactNode }) => {
  const isAuthenticated = !!localStorage.getItem('token');
  return isAuthenticated ? <>{children}</> : <Navigate to="/auth" />;
};

const App = () => {
  return (
    <ThemeProvider theme={theme}>
      <CssBaseline />
      <Router>
        <div className="app-container">
          <Navigation />
          <main className="main-content">
            <Routes>
              <Route path="/" element={<Home />} />
              <Route path="/services" element={<Services />} />
              <Route path="/chat" element={<Chat />} />
              <Route path="/auth" element={<Auth />} />
              <Route
                path="/profile"
                element={
                  <PrivateRoute>
                    <Profile />
                  </PrivateRoute>
                }
              />
              <Route
                path="/profile/edit"
                element={
                  <PrivateRoute>
                    <EditProfile />
                  </PrivateRoute>
                }
              />
              <Route
                path="/checkout"
                element={
                  <PrivateRoute>
                    <Checkout />
                  </PrivateRoute>
                }
              />
```

```
            </Routes>
          </main>
          <Footer />
        </div>
      </Router>
    </ThemeProvider>
  );
};

export default App;
```

ПРОГРАМНИЙ КОД СЕРВЕРНА ЧАСТИНА (BACKEND)

```
import express, { Request, Response, NextFunction } from 'express';
import cors from 'cors';
import { fileURLToPath } from 'url';
import { dirname } from 'path';
import jwt from 'jsonwebtoken';
import { register, login } from './auth/auth.js';
import { getDatabase } from './database/config.js';
import { join } from 'path';

const __filename = fileURLToPath(import.meta.url);
const __dirname = dirname(__filename);
const JWT_SECRET = process.env.JWT_SECRET || 'your-secret-key';

declare global {
  namespace Express {
    interface Request {
      user?: {
        userId: number;
      };
    }
  }
}

const app = express();
const port = process.env.PORT || 3001;

app.use(cors());
app.use(express.json());
const authenticateToken = (req: Request, res: Response, next: NextFunction) => {
  const authHeader = req.headers.authorization;
  const token = authHeader && authHeader.split(' ')[1];

  if (!token) {
    return res.status(401).json({ message: 'No token provided' });
  }

  jwt.verify(token, JWT_SECRET, (err: jwt.JsonWebTokenError | null, decoded: any) => {
    if (err) {
      return res.status(401).json({ message: 'Invalid token' });
    }
    req.user = decoded;
    next();
  });
};

app.post('/api/auth/register', async (req, res) => {
  try {
    const result = await register(req.body);
    res.json(result);
  } catch (error) {
    res.status(400).json({ message: error instanceof Error ? error.message : 'Registration
failed' });
  }
});

app.post('/api/auth/login', async (req, res) => {
  try {
    const result = await login(req.body);
    res.json(result);
  } catch (error) {
```

```
    res.status(401).json({ message: error instanceof Error ? error.message :
'Authentication failed' });
  }
});

app.get('/api/profile', authenticateToken, async (req: Request, res: Response) => {
  try {
    const db = await getDatabase();
    if (!req.user) {
      return res.status(401).json({ message: 'User not authenticated' });
    }

    const user = await db.get('SELECT id, firstName, lastName, email, role FROM users
WHERE id = ?', [req.user.userId]);

    if (!user) {
      return res.status(404).json({ message: 'User not found' });
    }

    const orders = await db.all('SELECT * FROM orders WHERE userId = ?', [user.id]);

    res.json({ user, orders });
  } catch (error) {
    res.status(500).json({ message: error instanceof Error ? error.message : 'Failed to
fetch profile' });
  }
});

app.put('/api/profile', authenticateToken, async (req: Request, res: Response) => {
  try {
    const db = await getDatabase();
    if (!req.user) {
      return res.status(401).json({ message: 'User not authenticated' });
    }

    const { firstName, lastName, email } = req.body;

    const existingUser = await db.get('SELECT id FROM users WHERE email = ? AND id != ?',
[email, req.user.userId]);
    if (existingUser) {
      return res.status(400).json({ message: 'Email already in use' });
    }

    await db.run(
      'UPDATE users SET firstName = ?, lastName = ?, email = ?, updatedAt =
CURRENT_TIMESTAMP WHERE id = ?',
      [firstName, lastName, email, req.user.userId]
    );

    const updatedUser = await db.get(
      'SELECT id, firstName, lastName, email, role FROM users WHERE id = ?',
      [req.user.userId]
    );

    res.json({ user: updatedUser });
  } catch (error) {
    res.status(500).json({ message: error instanceof Error ? error.message : 'Failed to
update profile' });
  }
});

app.post('/api/orders', authenticateToken, async (req: Request, res: Response) => {
```

```
  try {
    const db = await getDatabase();
    if (!req.user) {
      return res.status(401).json({ message: 'User not authenticated' });
    }

    const {
      projectName,
      phoneNumber,
      city,
      serviceId,
      additionalInfo,
      status,
      date
    } = req.body;

    const result = await db.run(
      `INSERT INTO orders (
        userId,
        serviceId,
        projectName,
        phoneNumber,
        city,
        additionalInfo,
        status,
        date
      ) VALUES (?, ?, ?, ?, ?, ?, ?, ?)`,
      [
        req.user.userId,
        serviceId,
        projectName,
        phoneNumber,
        city,
        additionalInfo,
        status,
        date
      ]
    );

    const order = await db.get(
      'SELECT * FROM orders WHERE id = ?',
      [result.lastID]
    );

    res.status(201).json({ order });
  } catch (error) {
    res.status(500).json({ message: error instanceof Error ? error.message : 'Failed to
create order' });
  }
});

app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

ПРОГРАМНИЙ КОД БАЗА ДАНИХ (DATABASE)

```
import sqlite3 from 'sqlite3';
import { open } from 'sqlite';
import { fileURLToPath } from 'url';
import { dirname, join } from 'path';

const __filename = fileURLToPath(import.meta.url);
const __dirname = dirname(__filename);
export const initializeDatabase = async () => {
  const db = await open({
    filename: join(__dirname, '../../../database.sqlite'),
    driver: sqlite3.Database
  });

  await db.exec(`
    CREATE TABLE IF NOT EXISTS users (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      firstName TEXT NOT NULL,
      lastName TEXT NOT NULL,
      email TEXT UNIQUE NOT NULL,
      password TEXT NOT NULL,
      role TEXT DEFAULT 'client',
      createdAt DATETIME DEFAULT CURRENT_TIMESTAMP,
      updatedAt DATETIME DEFAULT CURRENT_TIMESTAMP
    )
  `);

  await db.exec(`
    CREATE TABLE IF NOT EXISTS orders (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      userId INTEGER NOT NULL,
      serviceId INTEGER NOT NULL,
      projectName TEXT NOT NULL,
      phoneNumber TEXT NOT NULL,
      city TEXT NOT NULL,
      additionalInfo TEXT,
      status TEXT NOT NULL,
      date TEXT NOT NULL,
      FOREIGN KEY (userId) REFERENCES users(id)
    )
  `);
  return db;
};

let dbInstance: any = null;

export const getDatabase = async () => {
  if (!dbInstance) {
    dbInstance = await initializeDatabase();
  }
  return dbInstance;};
```

ПРОГРАМНИЙ КОД ЗВЕРНЕННЯ ДО AI

```
import axios from 'axios';
import { User, PRService, Order, ChatMessage } from '../types';

const API_URL = process.env.REACT_APP_API_URL || 'http://localhost:3001/api';

const api = axios.create({
  baseURL: API_URL,
  headers: {
    'Content-Type': 'application/json',
  },
});

// Auth API
export const authAPI = {
  login: async (email: string, password: string): Promise<User> => {
    const response = await api.post('/auth/login', { email, password });
    return response.data;
  },
  register: async (userData: Omit<User, 'id'>): Promise<User> => {
    const response = await api.post('/auth/register', userData);
    return response.data;
  },
  logout: async (): Promise<void> => {
    await api.post('/auth/logout');
  },
};

// Services API
export const servicesAPI = {
  getAllServices: async (): Promise<PRService[]> => {
    const response = await api.get('/services');
    return response.data;
  },
  getServiceById: async (id: string): Promise<PRService> => {
    const response = await api.get(`/services/${id}`);
    return response.data;
  },
  createOrder: async (orderData: Omit<Order, 'id' | 'createdAt' | 'updatedAt'>):
Promise<Order> => {
    const response = await api.post('/orders', orderData);
    return response.data;
  },
  getUserOrders: async (userId: string): Promise<Order[]> => {
    const response = await api.get(`/orders/user/${userId}`);
    return response.data;
  },
};

// Chat API
export const chatAPI = {
  sendMessage: async (message: Omit<ChatMessage, 'id' | 'timestamp'>):
Promise<ChatMessage> => {
    const response = await api.post('/chat/messages', message);
    return response.data;
  },
  getChatHistory: async (userId: string): Promise<ChatMessage[]> => {
    const response = await api.get(`/chat/history/${userId}`);
    return response.data;
  },
};
```

```
// User API
export const userAPI = {
  updateProfile: async (userId: string, userData: Partial<User>): Promise<User> => {
    const response = await api.put(`/users/${userId}`, userData);
    return response.data;
  },
  getProfile: async (userId: string): Promise<User> => {
    const response = await api.get(`/users/${userId}`);
    return response.data;
  },
};

// Interceptors
api.interceptors.request.use(
  (config) => {
    const token = localStorage.getItem('token');
    if (token) {
      config.headers.Authorization = `Bearer ${token}`;
    }
    return config;
  },
  (error) => {
    return Promise.reject(error);
  }
);

api.interceptors.response.use(
  (response) => response,
  (error) => {
    if (error.response?.status === 401) {
      localStorage.removeItem('token');
      window.location.href = '/login';
    }
    return Promise.reject(error);
  }
);

export default api;
```

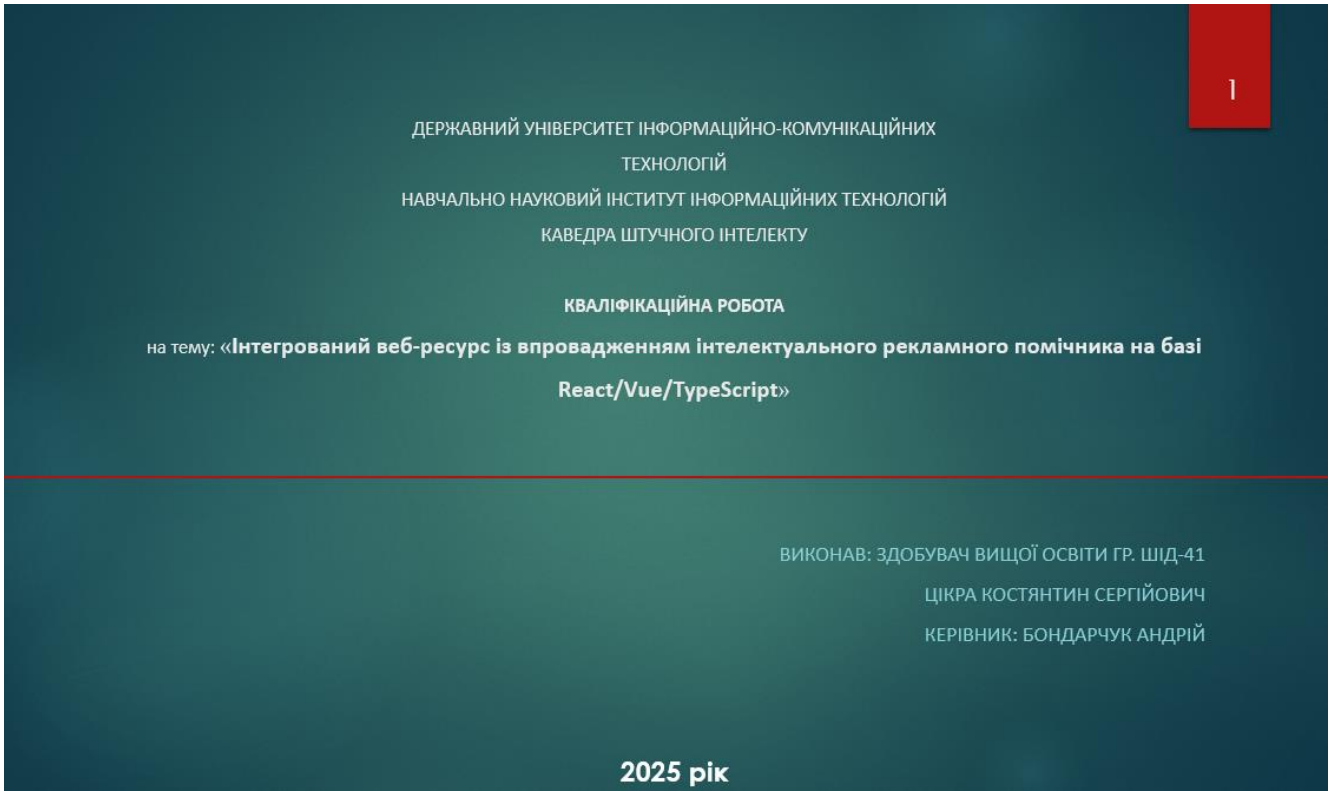
ПРОГРАМНИЙ КОД AI ФУНКЦІЙ

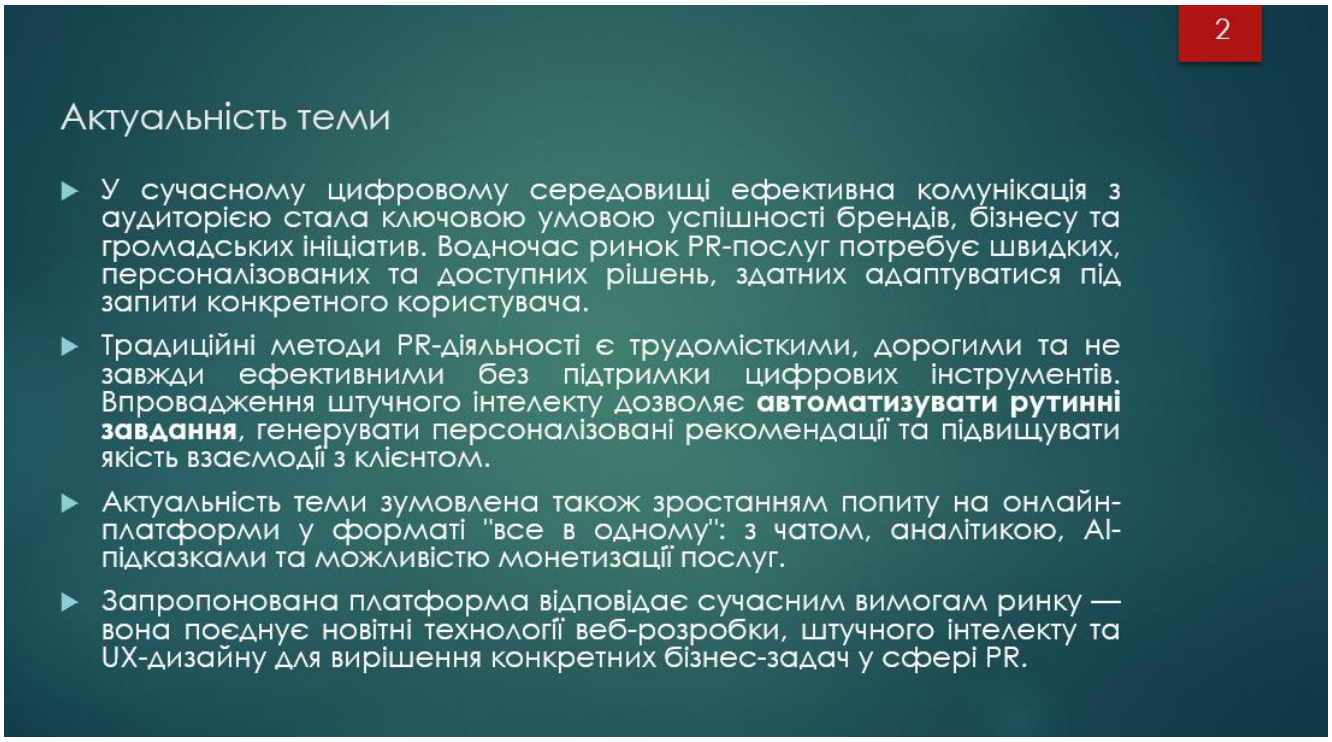
```
import { API_CONFIG } from '../config/api';

const INITIAL_CONTEXT = `You are an elite PR manager with over 10 years of experience in
managing high-profile clients and campaigns. You have successfully handled crisis
management, brand development, media relations, and strategic communications for Fortune
500 companies. You should respond as a seasoned PR professional, providing strategic,
practical, and industry-tested advice. Use your extensive experience to give detailed,
actionable insights and always maintain a professional, confident tone. Your responses
should reflect deep industry knowledge and best practices in modern PR and
communications.`;

interface GeminiResponse {
  candidates: Array<{
    content: {
      parts: Array<{
        text: string;
      }>;
    };
  }>;
}
export const generateAIResponse = async (message: string): Promise<string> => {
  try {
    const response = await
fetch(`${API_CONFIG.GEMINI_API_URL}?key=${API_CONFIG.GEMINI_API_KEY}`, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({
        contents: [{
          parts: [{
            text: `${INITIAL_CONTEXT}\n\nUser message: ${message}\n\nYour response as an
experienced PR manager:`
          }]
        }]
      })
    });

    if (!response.ok) {
      throw new Error('Failed to get response from Gemini API');
    }
    const data: GeminiResponse = await response.json();
    return data.candidates[0]?.content?.parts[0]?.text || 'Sorry, I could not generate a
response.';
  } catch (error) {
    console.error('Error generating AI response:', error);
    return 'Sorry, there was an error processing your request.';
  }};
```

ПРОГРАМНИЙ КОД ЗВЕРНЕНЯ JWT

```
import bcrypt from 'bcrypt';
import jwt from 'jsonwebtoken';
import { getDatabase } from '../database/config.js';

const JWT_SECRET = process.env.JWT_SECRET || 'your-secret-key';

interface RegisterData {
  firstName: string;
  lastName: string;
  email: string;
  password: string;
}
interface LoginData {
  email: string;
  password: string;
}
export const register = async (data: RegisterData) => {
  const db = await getDatabase();

  const existingUser = await db.get('SELECT * FROM users WHERE email = ?', [data.email]);
  if (existingUser) {
    throw new Error('User already exists');
  }
  const salt = await bcrypt.genSalt(10);
  const hashedPassword = await bcrypt.hash(data.password, salt);
  const result = await db.run(
    'INSERT INTO users (firstName, lastName, email, password) VALUES (?, ?, ?, ?)',
    [data.firstName, data.lastName, data.email, hashedPassword]
  );
  const token = jwt.sign({ userId: result.lastID }, JWT_SECRET, { expiresIn: '24h' });

  return { token };
};

export const login = async (data: LoginData) => {
  const db = await getDatabase();

  const user = await db.get('SELECT * FROM users WHERE email = ?', [data.email]);
  if (!user) {
    throw new Error('Invalid credentials');
  }

  const validPassword = await bcrypt.compare(data.password, user.password);
  if (!validPassword) {
    throw new Error('Invalid credentials');
  }

  const token = jwt.sign({ userId: user.id }, JWT_SECRET, { expiresIn: '24h' });

  return { token };
};

export const verifyToken = (token: string) => {
  try {
    return jwt.verify(token, JWT_SECRET);
  } catch (error) {
    throw new Error('Invalid token');
  }
};
```

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ





Постановка задачі

- **Мета роботи:**
  розробка веб-платформи для надання персоналізованих PR-послуг з використанням сучасних веб-технологій і інтеграцією інтелектуального помічника на основі штучного інтелекту.
- **Об'єкт дослідження:**
  процеси автоматизації комунікацій у сфері цифрового PR.
- **Предмет дослідження:**
  засоби створення веб-застосунків, а також методи генерації рекомендацій PR-контенту за допомогою мовних моделей (GPT).
- **Основні завдання кваліфікаційної роботи:**

1. Розробка архітектури платформи з клієнтською та серверною частиною.
2. Інтеграція інтелектуального помічника для формування PR-рекомендацій.
3. Реалізація системи реального часу та обробки платежів.

Опис процесу роботи платформи

•Реєстрація користувача

Користувач створює обліковий запис або входить у систему для доступу до функціоналу платформи.

•Створення нової PR-кампанії

Через зручний інтерфейс користувач заповнює форму з деталями кампанії: тема, ціль, аудиторія тощо.

•Запит до інтелектуального помічника

Після збереження кампанії система автоматично надсилає запит до AI-помічника для генерації персоналізованих рекомендацій.

•Отримання PR-рекомендацій

Помічник формує пропозиції щодо заголовків, каналів комунікації, ключових повідомлень, формату контенту тощо.

•Редагування та збереження результату

Користувач переглядає відповіді, обирає релевантні варіанти та зберігає їх до кампанії для подальшої реалізації.

•Спілкування через чат

За потреби користувач може вести діалог з AI або оператором у чаті в режимі реального часу (Socket.io).

•Оплата та аналітика

Доступ до розширених функцій відкривається після оплати через Stripe. Платформа надає статистику результатів кампанії.

Опис інтелектуального помічника GPT

- Інтелектуальний помічник платформи заснований на **мовній моделі GPT (Generative Pre-trained Transformer)**, що є прикладом сучасного застосування штучного інтелекту у сфері текстової генерації.
- Модель попередньо навчена на великих обсягах текстових даних і здатна **аналізувати природну мову**, формувати осмислені відповіді, а також **генерувати персоналізовані PR-рекомендації**.
- Помічник враховує тему кампанії, цільову аудиторію та попередню історію користувача. У відповідь він може запропонувати:
  – заголовки для публікацій;
  – ключові меседжі та ідеї контенту;
  – вибір каналів комунікації;
  – короткий план PR-дій.
- GPT інтегровано в платформу через REST API та реагує в режимі реального часу у чаті користувача.

Процес генерації сценаріїв

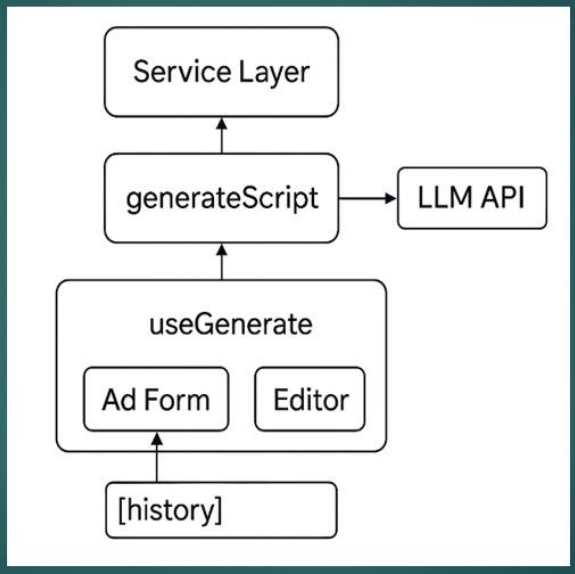


Рис. 1 Структурна схема генерації сценаріїв

Принцип роботи системи

- Принцип роботи платформи ґрунтується на взаємодії клієнтської частини, серверної логіки та штучного інтелекту в межах веб-застосунку зробленого за допомогою React.

1. **Користувач створює кампанію**, вказуючи її тему, мету, цільову аудиторію та інші параметри.
2. **Ці дані надсилаються на сервер**, де відбувається попередня обробка.
3. **Сервер формує запит до AI-помічника (GPT)** через REST API з урахуванням контексту кампанії.
4. **AI генерує PR-рекомендації**, такі як заголовки, меседжі, пропозиції каналів просування.
5. **Результати повертаються до клієнта** та виводяться у чаті в режимі реального часу (Socket.io).
6. Користувач може **редагувати, зберегти або відправити** рекомендації в роботу.
7. Доступ до розширених функцій відкривається після оплати через Stripe, який функціонує за допомогою Vue.js.

Використані елементи

•**React+TypeScript**
Використано для створення динамічного інтерфейсу користувача з типізацією та підтримкою SPA-архітектури.
•**Node.js+Express**
Серверна частина реалізована з використанням Express — легкого фреймворку для побудови REST API.
•**PostgreSQL**
Реляційна база даних для зберігання інформації про користувачів, кампанії, повідомлення та платежі.
•**Socket.io**
Забезпечує двосторонню комунікацію в реальному часі між користувачем і системою (чат, помічник).
•**OpenAI API (GPT)**
Інтегрована мовна модель для генерації персоналізованих PR-рекомендацій у чаті.





Головне меню:

Коли ви пройдете реєстрацію, попадете на головну сторінку де зможете подивитися всю інформацію про проєкт та вибрати або чат з AI, або перейти відразу до вибору сервісу

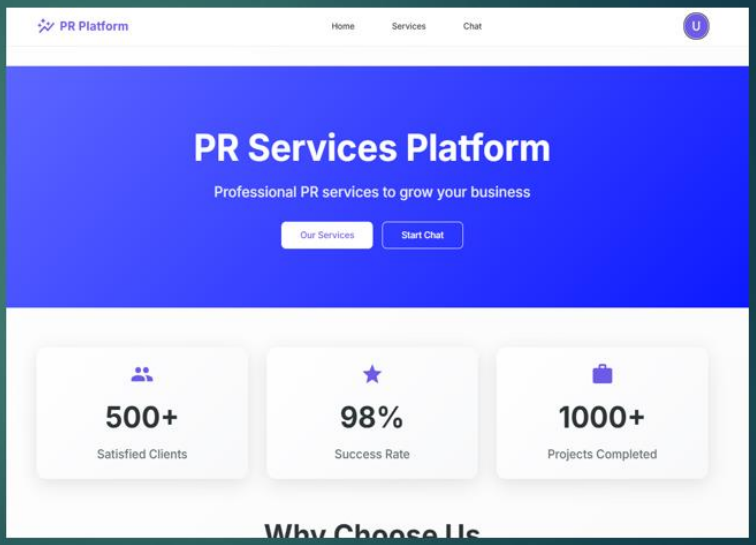


Рис. 2 Головне меню

Чат з AI помічником:

Тут ви можете з ним порадитись яку сервісну піар кампанію слід краще підібрати під ваш проєкт, у ньому по стандарту стоїть вписаний промпт, який завдає йому базові характеристики піар менеджера. Такий підхід можна використовувати лише з новими моделями.

Рис. 3 Чат з AI помічником

Вибір сервісу для піар кампанії:

Ви вибираєте той сервіс який порекомендує вам AI помічник, у кожного сервісу свій підхід та ціна. Коли ви нажимаєте використати цей шаблон, вас перекине на сторінку де ви вписуєте всю інформацію про свій проект.

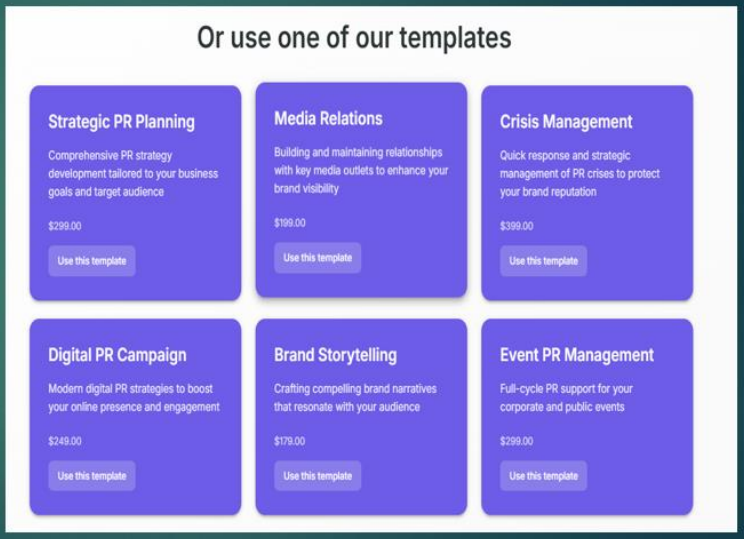


Рис. 4 Сервіси та кампанії

Висновок

- Розроблено веб-платформу для автоматизації PR-послуг із використанням сучасних веб-технологій і мовної моделі штучного інтелекту.
- Основними завданнями були розробка клієнтської та серверної частини, інтеграція інтелектуального помічника, реалізація чату та платіжної системи.
- Інтелектуальний помічник створено на основі GPT-моделі, яка формує рекомендації для PR-кампаній на основі запитів користувача.
- Реалізовано зручний адаптивний інтерфейс із підтримкою SPA, що дозволяє користувачам створювати й керувати кампаніями, отримувати поради й аналітику в одному середовищі.
- Отриманий продукт забезпечує ефективне управління PR-діяльністю, автоматизацію рутинних процесів і можливість масштабування для комерційного використання.