НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО **Декан факультету**	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ** **Завідувач кафедри інформаційних систем і технологій**
_____ **Ігор БОЛБОТ** (підпис)	_____ **Михайло ШВИДЕНКО** (підпис)
«___» _____ 2026 р.	«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Веб-орієнтована система для моніторингу ефективності рекламних компаній в соціальній мережі TikTok**»**
Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Гарант освітньої програми
д.е.н., професор _____ **Роман РУДЕНСЬКИЙ**
(підпис)

Керівник бакалаврської кваліфікаційної роботи
доктор філософії, доцент _____ **Валентина КОРОЛЬЧУК**
(підпис)

Виконав _____ **Тарас БЕЗУШКО**
(підпис)

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри інформаційних систем і технологій

к.е.н., доцент _____ Михайло ШВИДЕНКО
(підпис)

«___» _____ 2026 р.

ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ

Безушку Тарасу Олеговичу
(прізвище, ім'я, по батькові)

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Тема бакалаврської кваліфікаційної роботи

«Веб-орієнтована система для моніторингу ефективності рекламних компаній в соціальній мережі TikTok»

затверджена наказом від «06» січня 2026 р. № 46 «С»
Термін подання завершеної роботи на кафедру «___» _____ 2026 р.
Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту): аналітичні звіти, статистичні показники рекламних кампаній, візуалізація ефективності реклами, результати моніторингу аудиторії, рекомендації щодо оптимізації рекламних кампаній та експортовані звіти у різних форматах.
Перелік питань, що підлягають дослідженню:

1. Аналіз сучасних підходів до цифрового маркетингу та рекламних кампаній у соціальних мережах.
2. Дослідження особливостей функціонування та рекламних механізмів соціальної мережі TikTok.
3. Аналіз існуючих веб-орієнтованих систем моніторингу та аналітики рекламних кампаній.
4. Визначення ключових показників ефективності рекламних кампаній у TikTok.
5. Аналіз API та інструментів інтеграції платформи TikTok для отримання маркетингових даних.
6. Проєктування архітектури веб-орієнтованої системи моніторингу ефективності реклами.
7. Розроблення структури бази даних для зберігання статистики та аналітичної

інформації.
8. Реалізація функціоналу збору, аналізу та автоматизованого оновлення даних рекламних кампаній.
9. Розроблення користувацького веб-інтерфейсу для відображення аналітичної інформації та звітності.
10. Забезпечення безпеки, надійності та продуктивності веб-орієнтованої системи.
11. Проведення тестування системи та оцінювання її ефективності в умовах реального використання.

Дата видачі завдання «19» січня 2026 р.

Керівник бакалаврської кваліфікаційної роботи _____ (підпис) **Валентина КОРОЛЬЧУК**

Завдання взяв до виконання _____ (підпис) **Тарас БЕЗУШКО**

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – інтерфейс прикладного програмування.

BPMN – нотація та модель бізнес-процесів.

Burnout Rate – показник рівня вигорання рекламних креативів.

CSRF – підробка міжсайтових запитів.

CTA – заклик до дії у рекламному відеоролику.

CTR – показник клікабельності реклами.

DNS – система доменних імен.

DOM – об’єктна модель документа.

HTTP / HTTPS – протокол передачі гіпертексту / безпечний протокол передачі гіпертексту.

JSON – текстовий формат обміну даними.

KPI – ключовий показник ефективності.

MVT – архітектура «модель–представлення–шаблон»

OAuth 2.0 – відкритий протокол авторизації без передачі пароля.

ORM – об’єктно-реляційне відображення бази даних.

RBAC – контроль доступу на основі ролей. ROI – показник рентабельності інвестицій.

ROI – показник рентабельності інвестицій.

SQL – мова структурованих запитів до бази даних.

SSL/TLS – криптографічні протоколи для безпечного мережевого зв'язку.

UML – уніфікована мова моделювання програмних систем.

UX/UI – користувацький досвід та інтерфейс користувача.

XSS – міжсайтовий скриптинг (вразливість веб-систем).

СУБД – система управління базами даних.

ТЗ – технічне завдання.

ВСТУП

Стрімкий розвиток електронної комерції та соціальних платформ кардинально змінив підходи до цифрового маркетингу, де сучасний рекламний ринок вимагає від компаній вміння миттєво привертати увагу цільової аудиторії, а платформа TikTok завдяки специфіці своїх алгоритмів та акценту на короткому вертикальному відеоконтенті стала одним із найвпливовіших каналів залучення трафіку. Специфіка цієї мережі полягає у безперервній ротації трендів, що змушує маркетингові агенції працювати в умовах надзвичайно стислих термінів і генерувати величезну кількість матеріалів щотижня. Особливу складність у цьому процесі становить явище швидкого вигорання рекламних креативів, коли через часті покази ролика його ефективність стрімко падає, відображаючись у зниженні клікабельності та зростанні вартості залучення клієнта, що веде до прямої втрати бюджету фахівцями із закупівлі реклами. Це зумовлює безперервну потребу в аналізі фінансових показників кожного окремого відео та оперативному замовленні нових матеріалів у дизайнерському відділі, проте традиційна координація між аналітиками та креативниками часто стає найвужчим місцем у всьому виробничому ланцюжку.

Звичайні підходи до управління такими кампаніями, що спираються на базові можливості стандартних рекламних кабінетів або ж на розрізнені електронні таблиці та чати, виявляються недостатніми для великих команд, оскільки аналітичні дані доводиться збирати вручну, а передача завдань супроводжується втратою інформації та непорозуміннями щодо причин відхилення попередніх робіт. Створення спеціалізованого програмного середовища здатне повністю трансформувати цей підхід завдяки автоматизованому збору метрик через програмні інтерфейси соціальних мереж та інтеграції фінансової аналітики з модулем постановки завдань в одному інтерфейсі. Дизайнер при цьому одразу бачить детальне технічне завдання з необхідними параметрами тривалості та текстових зачіпок, а замовник може контролювати статус виконання роботи, що мінімізує людський фактор, прискорює реакцію на зміну ринкових умов та захищає ресурси від

неефективного використання.

Головна ціль цієї кваліфікаційної роботи зводиться до проєктування та практичної розробки веб-орієнтованої інформаційної системи, призначеної для автоматизованого моніторингу метрик ефективності в мережі TikTok та оптимізації взаємодії всередині маркетингових команд, де об'єктом дослідження виступають процеси організації, аналізу та моніторингу рекламних активностей у цифровому просторі соціальних мереж. Предметом дослідження є моделі, архітектурні патерни, алгоритми та програмні інструменти автоматизації агрегації аналітичних даних і прозорого управління життєвим циклом рекламних креативів на базі веб-технологій. Для вирішення поставлених завдань, що охоплюють увесь повний цикл від аналізу предметної області, проєктування логічної структури бази даних, програмної реалізації аналітичного двигуна обчислення коефіцієнтів рентабельності до комплексного функціонального тестування та інфраструктурного налаштування, було використано методи системного аналізу, принципи об’єктно-орієнтованого проєктування, теорію реляційних баз даних, підходи математичного моделювання, а також сучасні методології профілювання швидкодії програмного забезпечення.

Результати роботи мають високу практичну цінність для сучасних маркетингових агенцій, оскільки розроблені архітектурні рішення дозволяють замкнути цикл виробництва контенту в єдиному відмовостійкому інтерфейсі та забезпечити точний автоматичний розрахунок показників ROI, CTR та Burnout Rate.

Наукова та практична зрілість створеного програмного комплексу була підтверджена під час апробації результатів дослідження, де основні теоретичні положення були представлені та обговорені на VII Всеукраїнській науково-практичній конференції студентів та аспірантів «Теоретичні та прикладні аспекти розробки комп’ютерних систем '2026», за результатами якої опубліковано тези доповіді на тему «Веб-орієнтована система для моніторингу ефективності рекламних компаній в соціальній мережі TikTok». Такий комплексний підхід дозволив створити масштабоване технологічне рішення,

повністю готове до промислового впровадження у реальному секторі цифрового маркетингу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Аналіз TikTok як платформи для просування товарів та послуг

Сучасний ландшафт цифрового маркетингу зазнав кардинальних трансформацій з появою та стрімким розвитком соціальної мережі TikTok. Ця платформа за досить короткий проміжок часу перетворилася з майданчика для розважальних відеороликів на потужну глобальну екосистему для просування брендів і товарів. Головною особливістю платформи є її унікальний підхід до розповсюдження інформації, який докорінно відрізняється від традиційних соціальних медіа попереднього покоління. Якщо більшість існуючих мереж базується на соціальному графі, де видимість контенту безпосередньо залежить від кількості підписників та сталих зв'язків між користувачами, то TikTok функціонує на основі графа інтересів. Така архітектура дозволяє алгоритмам фокусуватися на поведінкових факторах конкретної людини, детально аналізуючи кожну секунду перегляду та глибину взаємодії з кожним окремим роликом.

Механізм роботи алгоритмів забезпечує відносно рівні умови для всіх учасників рекламного ринку, незалежно від масштабу їхньої діяльності чи наявності великої бази лояльних клієнтів. Кожне завантажене відео проходить через систему автоматизованого багаторівневого тестування, де на початковому етапі воно демонструється невеликій групі випадкових користувачів із потенційно схожими вподобаннями. Згідно з офіційною інформацією розробників платформи щодо принципів роботи рекомендаційної стрічки, система безперервно зважує такі фактори, як рівень утримання уваги, частота вподобань, коментарі та навіть те, чи додивився користувач відео до кінця, що у сукупності створює унікальний ефект віральності, який неможливо гарантувати лише обсягом вкладених фінансових коштів у налаштування таргетингу [2]. Для

бізнесу це означає, що успіх будь-якої рекламної кампанії прямо залежить від здатності контенту зацікавити глядача з перших миттєвостей перегляду.

Для реалізації комерційних цілей та залучення цільового трафіку платформа пропонує гнучкий набір спеціалізованих рекламних форматів, серед яких базовим вважається In-Feed Ads. Цей формат передбачає розміщення рекламного відеоролика безпосередньо у стрічці рекомендацій користувача між органічними публікаціями інших авторів. Головною перевагою In-Feed Ads є їхня максимальна нативність, оскільки вони виглядають як частина природного потоку відеоконтенту, що значно знижує рівень роздратування аудиторії від прямої комерційної пропозиції. Користувач має змогу взаємодіяти з такою рекламою так само, як і з будь-яким іншим відео: ставити вподобання, залишати коментарі, переглядати профіль або поширювати матеріал серед знайомих. Кожна така публікація обов'язково містить інтерактивну кнопку із закликом до дії, що дозволяє миттєво перенаправити зацікавленого користувача на зовнішній сайт чи сторінку завантаження додатка.

Ще більш інноваційним та стратегічно важливим інструментом є формат Spark Ads, який концептуально змінює підхід до платного просування, дозволяючи брендам використовувати вже існуючі органічні публікації як власного профілю, так і контент інших креаторів за їхньої згоди. Цей формат забезпечує надзвичайно високу автентичність маркетингового повідомлення, адже реклама сприймається аудиторією як щира рекомендація від живої людини, при цьому всі реакції, перегляди та коментарі назавжди залишаються на оригінальному пості, що стимулює довгострокове органічне зростання профілю бренду та підвищує його загальну впізнаваність [1]. Такий підхід ідеально відповідає загальній філософії платформи, яка вимагає від бізнесу створювати цінний розважальний контент, а не просто переривати перегляд користувача стандартними рекламними банерами.

Специфіка алгоритмічного просування в середовищі коротких відео створює умови, за яких будь-які рекламні матеріали мають дуже обмежений термін високої ефективності. Якщо у класичних медіа одна вдала рекламна ідея могла стабільно працювати протягом багатьох місяців, то тут користувачі

надзвичайно швидко звикають до візуальних образів, що призводить до їхнього автоматичного ігнорування під час прокручування стрічки. Це спричиняє об'єктивну необхідність постійного оновлення візуальних креативів та проведення безперервних тестів різних підходів до подачі інформації. Команди медіабайїнгу вимушені виробляти контент практично у промислових масштабах, де кожен ролик перевіряє певну гіпотезу щодо поведінки та реакцій цільової аудиторії. Саме ця особливість середовища визначає критичну складність процесів моніторингу метрик, оскільки найменша затримка в аналізі актуальних цифр неминуче призводить до марного витрачання бюджету на контент, який вже втратив свою привабливість.

З технологічної точки зору, рекламна інфраструктура платформи є складною системою обробки великих даних, де кожна дія користувача миттєво стає сигналом для нейронної мережі. Процес закупівлі показів у такій системі відбувається за моделлю динамічного аукціону, де перемагає не лише рекламодавець із найвищою ставкою, а й оголошення з найвищим прогнозованим рівнем залученості. Це означає, що система економічно заохочує той контент, який користувачі доглядають до кінця та на який вони найактивніше реагують. Для успішного функціонування в таких висококонкурентних умовах маркетинговим фахівцям необхідно мати безперебійний доступ до детальної аналітики в режимі реального часу, щоб розуміти динаміку змін фінансових показників та вчасно масштабувати найуспішніші зв'язки.

На завершення цього аналізу варто підкреслити, що успішне використання платформи вимагає від компаній глибокого розуміння її поточної культури та постійно мінливих трендів. Рекламні кампанії, що ігнорують специфічний стиль платформи та намагаються використовувати застарілі телевізійні підходи, зазвичай демонструють вкрай низькі фінансові результати, незалежно від реальної якості самого продукту. Можливість безшовної інтеграції комерційних цілей у суто розважальний формат є головним викликом для розробників рекламних стратегій. Використання нативних форматів дозволяє створити комфортний досвід для користувача, де межа між розвагою та покупкою стає майже непомітною. Проте стабільна та прибуткова робота з цими форматами на

великих обсягах трафіку абсолютно неможлива без глибокої автоматизації збору статистичних даних та систематизації процесів виробництва креативів, що і зумовлює гостру необхідність розробки спеціалізованих інформаційних систем управління.

1.2. Дослідження проблем моніторингу рекламної ефективності

Основою успішного управління рекламними кампаніями в цифровому середовищі є безперервний аналіз ключових показників ефективності. У сфері таргетованої реклами найважливішими індикаторами результативності виступають коефіцієнт клікабельності та загальна рентабельність інвестицій. Коефіцієнт клікабельності відображає відсоткове відношення кількості користувачів, які здійснили перехід за рекламним посиланням, до загальної кількості переглядів відеоролика. Цей показник є первинним мірилом того, наскільки візуальна складова контенту та перші секунди відео виявилися привабливими для цільової аудиторії. Своєю чергою, рентабельність інвестицій є кінцевим фінансовим критерієм успіху, який демонструє співвідношення отриманого чистого прибутку до загальних витрат на кампанію. Для фахівця із закупівлі реклами моніторинг цих двох метрик у реальному часі є фундаментальним завданням, оскільки будь-яке критичне відхилення від норми сигналізує про необхідність термінового втручання в налаштування.

Специфіка платформи TikTok генерує унікальну проблему, яка практично не зустрічається у таких масштабах на інших класичних майданчиках. Йдеться про феномен швидкого вигорання рекламних креативів, відомий у професійному середовищі як Ad Burnout. Вигорання контенту визначається як стан, за якого аудиторія починає ігнорувати рекламне повідомлення через його надмірну частоту показів або швидку втрату ефекту новизни. Рекламні кампанії у цій соціальній мережі часто демонструють високу рентабельність у перші 24 чи 48 годин, але можуть стрімко вигоріти і перестати приносити прибуток вже на третій або п'ятий день активної роботи [3]. Це відбувається через те, що алгоритми платформи функціонують як високошвидкісний ринок уваги, де

система безперервно тестує новий контент і миттєво знижує пріоритет тих відеороликів, які перестають генерувати сильні поведінкові сигнали від користувачів [3].

На графіках аналітичних систем процес вигорання виглядає як різке падіння показника CTR при одночасному неконтрольованому зростанні вартості залучення одного клієнта. Проблема полягає в тому, що алгоритми соціальної мережі продовжують витрачати денні ліміти рекламодавців дуже високими темпами навіть у фазі спаду, коли відео вже перестало конвертувати глядачів у реальних покупців [2]. Медіабаєр змушений працювати в умовах постійного інформаційного навантаження, безперервно оновлюючи сторінки рекламних кабінетів, щоб встигнути зупинити неефективні кампанії до того, як вони завдадуть непоправної шкоди загальному ROI проєкту. Проте проста зупинка збиткової реклами є лише першим кроком, після якого починається найскладніший організаційний етап, пов'язаний із розробкою нового візуального контенту.

Саме на етапі оновлення креативів виникає критична проблема комунікації між двома абсолютно різними підрозділами компанії. З одного боку знаходяться фахівці з медіабаїнгу, які оперують виключно цифрами та відсотками конверсій, а з іншого боку працюють дизайнери, для яких пріоритетом виступає візуальна естетика та динаміка кадру. На практиці координація між цими відділами часто перетворюється на несистематизований обмін повідомленнями, що створює серйозні затримки у виробництві. Баєри формулюють технічні завдання безсистемно, часто не пояснюючи, чому саме попереднє відео вигоріло та які саме візуальні тригери потрібно змінити для відновлення високого CTR. Дизайнери, не маючи прямого доступу до аналітичної статистики, не розуміють об'єктивних причин падіння ефективності їхніх робіт. Така розрізненість інформаційного поля призводить до того, що нові відео створюються наосліп, а рекламні кампанії простоюють в очікуванні свіжих матеріалів, що зрештою призводить до втрати потенційного прибутку.

1.3. Огляд існуючих систем аналітики та обґрунтування розробки власного рішення

Вибір ефективного інструментарію для моніторингу рекламних кампаній є критичним етапом в організації роботи сучасної маркетингової команди. На сьогодні ринок пропонує широкий спектр рішень, які можна умовно розділити на три категорії: нативні кабінети соціальних мереж, професійні аналітичні платформи та універсальні системи управління проєктами. Кожне з цих рішень має свої технологічні переваги, проте вони не позбавлені суттєвих недоліків у контексті специфіки роботи з TikTok. Найбільш очевидним вибором для будь-якого рекламодавця є офіційний кабінет TikTok Ads Manager. Цей інструмент надає вичерпну інформацію про стан рекламних акаунтів, дозволяє відстежувати витрати бюджету, кількість показів та базові конверсії безпосередньо в інтерфейсі платформи [4]. Основною перевагою нативного рішення є максимальна точність даних, оскільки вони надходять безпосередньо з серверів соціальної мережі без затримок. Проте кабінет розрахований на індивідуальне управління кампаніями і не містить жодного функціоналу для координації роботи між баєром та дизайнером, що створює інформаційний вакуум на етапі підготовки креативів.

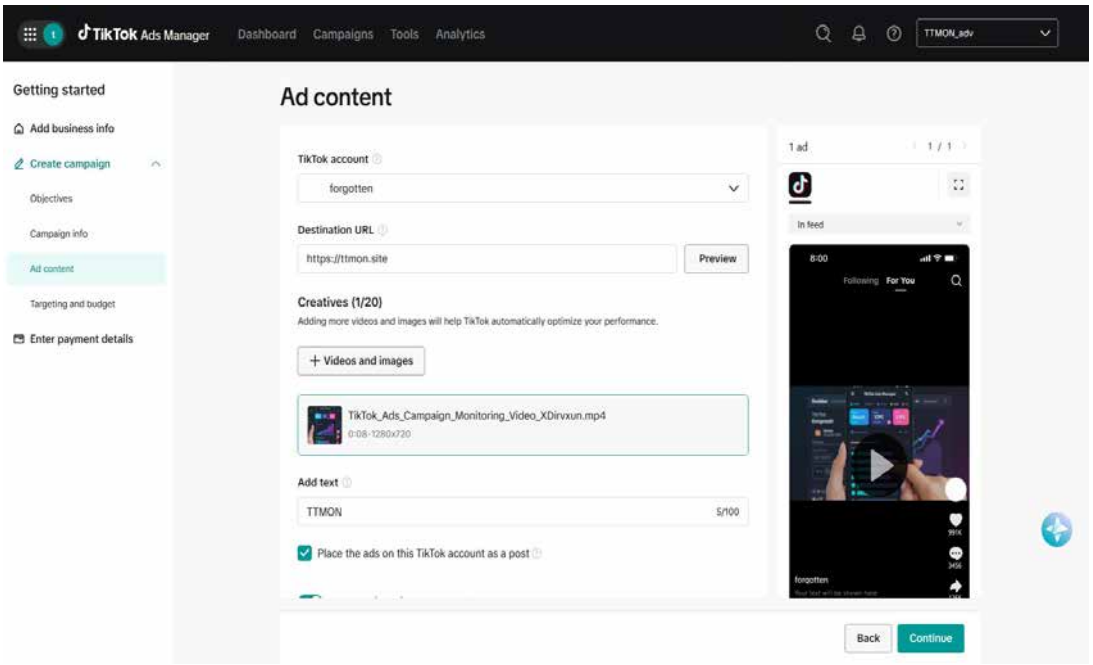

Рис. 1.1 – Інтерфейс рекламного кабінета TikTok Ads Manager при створенні рекламної кампанії

Для компаній, які працюють з великими обсягами мобільного трафіку та потребують глибшого аналізу поведінки користувачів, золотим стандартом є платформа AppsFlyer. Ця система дозволяє реалізувати наскрізну аналітику, відстежуючи шлях користувача від першого кліку по рекламному відео до здійснення цільової дії всередині мобільного додатка. Використання професійної атрибуції допомагає бізнесу об'єктивно оцінювати цінність кожного рекламного каналу та оптимізувати маркетингові витрати на основі реальних фінансових результатів [5]. Незважаючи на величезну аналітичну потужність, AppsFlyer є занадто складним та дорогим інструментом для команд, чия щоденна робота зосереджена на швидкій ротації креативів. Платформа не пропонує механізмів для візуального перегляду відеороликів або постановки технічних завдань, що змушує команди використовувати додаткові сторонні сервіси для управління виробничим циклом.

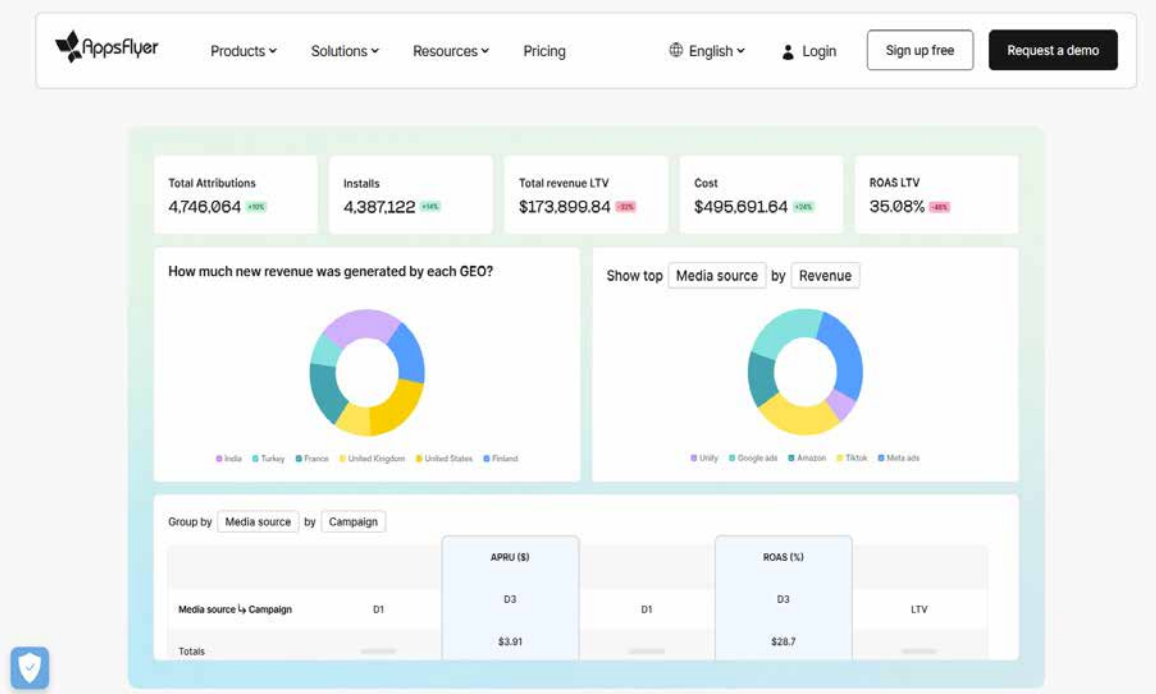

Рис. 1.2 – Сторінка дашбордів платформи AppsFlyer

Коли мова йде про організацію командної роботи та контроль за виконанням завдань, більшість компаній обирає систему Jira. Це універсальний інструмент, який дозволяє гнучко налаштовувати робочі процеси, створювати картки завдань, призначати відповідальних та відстежувати терміни виконання [6]. Для технічного відділу чи команди розробників Jira є ідеальним рішенням, проте у маркетинговому середовищі її використання часто виявляється надлишковим.

Головна проблема полягає у повній відсутності зв'язку між таск-трекером та реальною статистикою з рекламних кабінетів. Дизайнер, отримуючи завдання в Jira, бачить лише текстовий опис, але не має доступу до даних про те, як спрацювали його попередні відео. Це призводить до ситуації, коли креативний відділ працює у відриві від фінансових результатів, що негативно впливає на загальну ефективність рекламних стратегій.

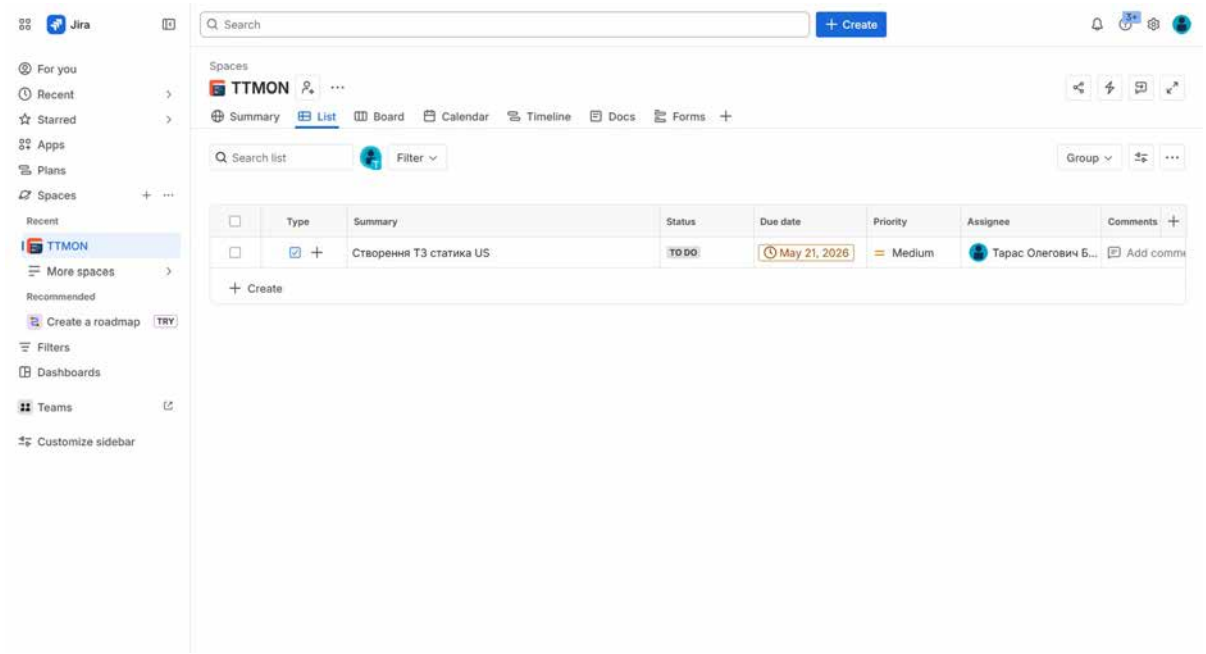

Рис. 1.3 – Робоче середовище сервісу управління проєктами Jira

Для наочного порівняння можливостей існуючих систем та обґрунтування необхідності розробки власного рішення TTMON було складено порівняльну характеристику за ключовими параметрами.

Таблиця 1.1

Порівняльний аналіз систем аналітики

Параметр порівняння	**TikTok Ads Manager**	**AppsFlyer**	**Jira Software**	**TTMON (проєкт)**
Пряма інтеграція з TikTok API	Повна нативна	Через SDK	Відсутня	Повна через API
Моніторинг фінансових метрик	Базовий	Поглиблений	Відсутній	Автоматизований
Управління технічними завданнями	Відсутнє	Відсутнє	Розширене	Спеціалізоване
Перегляд медіаконтенту	Тільки активні	Відсутній	Через вкладення	Вбудований плеєр
Аналіз "вигорання" креативів	Ручний аналіз	Автоматизований	Відсутній	Розумні тригери
Командна рольова модель	Обмежена	Складна	Гнучка	Оптимізована

Аналіз наведених даних підтверджує, що жодне з існуючих рішень не покриває одночасно потреби у фінансовій аналітиці та управлінні життєвим циклом креативів. TikTok Ads Manager та AppsFlyer фокусуються на цифрах, повністю ігноруючи процес створення контенту, тоді як Jira забезпечує лише контроль за завданнями без прив'язки до їхньої ринкової ефективності. Це створює серйозний комунікаційний розрив, який стає причиною втрати часу та рекламних бюджетів маркетингових агенцій. Баєри змушені вручну переносити статистику з однієї системи в іншу, щоб пояснити дизайнерам необхідність змін у відеороликах, що підвищує ризик виникнення помилок та уповільнює реакцію на зміни ринкових умов.

Розробка власної системи TTMON обґрунтовується необхідністю створення єдиного інформаційного простору, де кожне технічне завдання буде нерозривно пов'язане з реальною статистикою ефективності. Впровадження такого рішення дозволить автоматизувати процес виявлення ознак вигорання контенту та миттєво сигналізувати команді про потребу в оновленні матеріалів. Креативний відділ отримає доступ до об'єктивних даних про успішність своїх робіт, що стимулюватиме створення більш якісного контенту. Таким чином, власна розробка є логічним кроком до оптимізації внутрішніх витрат та підвищення прибутковості компанії через усунення розрізненості інструментів та покращення взаємодії всередині команди.

Ця система не намагається повністю замінити професійні інструменти глибокої аналітики, а скоріше виступає сполучною ланкою, яка адаптує складні технічні дані для щоденних потреб маркетингової команди. Використання кастомного рішення дозволяє реалізувати специфічні алгоритми розрахунку ROI, які враховують не лише витрати на рекламу, а й внутрішні витрати на виробництво кожного окремого креативу. Це забезпечує керівництву агенції повну прозорість бізнес-процесів та дозволяє приймати стратегічні рішення на основі консолідованої інформації, що раніше була розпорошена між багатьма непов'язаними сервісами.

1.4. Формулювання функціональних та технічних вимог до системи

Процес формування вимог є фундаментом розробки будь-якого програмного продукту, оскільки він визначає межі системи та критерії її успішної реалізації. Згідно з міжнародним стандартом ISO/IEC/IEEE 29148:2018, якісні вимоги повинні бути специфікованими, вимірними та досяжними, що дозволяє уникнути невизначеності на етапах проєктування та програмування [7]. Для системи моніторингу TTMON вимоги поділяються на декілька ключових категорій, які охоплюють бізнес-логіку, технічні обмеження та аспекти безпеки.

Функціональні вимоги визначають безпосередні дії, які система повинна виконувати для задоволення потреб користувачів. Вони охоплюють три основні модулі: аналітику, управління завданнями та адміністративну частину. Основні функціональні вимоги включають:

- Автоматизований збір даних з TikTok Marketing API щодо витрат, показів та конверсій у розрізі окремих рекламних кампаній та креативів.
- Динамічний розрахунок показників ефективності, таких як ROI та CTR, з можливістю фільтрації за часовими інтервалами та конкретними медіабаєрами.
- Систему сповіщень про виявлення ознак вигорання контенту на основі встановлених порогових значень падіння метрик.
- Функціонал створення технічних завдань для дизайнерів, що включає опис візуальних елементів, вимоги до тексту та тривалості відео.
- Модуль управління медіафайлами з можливістю попереднього перегляду їх безпосередньо в інтерфейсі системи.
- Механізм коментування та затвердження результатів роботи дизайнерами для забезпечення прозорого циклу виробництва контенту.

Технічні вимоги визначають програмне та апаратне середовище, у якому буде функціонувати додаток. Система повинна бути побудована за архітектурою

клієнт-сервер із використанням сучасних вебтехнологій. Основні технічні параметри:

- Використання мови програмування Python та фреймворку Django як базової платформи для бекенд-розробки.
- Застосування реляційної СУБД PostgreSQL для зберігання аналітичних даних та структурованої інформації про користувачів.
- Реалізація протоколу OAuth 2.0 для безпечної авторизації та взаємодії з програмними інтерфейсами TikTok.
- Забезпечення адаптивності інтерфейсу для коректного відображення на різних типах пристроїв через використання сучасних CSS фреймворків.
- Підтримка змінних середовища (.env) для зберігання секретних ключів, токенів доступу та параметрів підключення до бази даних.

Вимоги до ролей користувачів та моделі доступу ґрунтуються на принципі мінімальних привілеїв. Система повинна підтримувати модель управління доступом на основі ролей, що передбачає наступні рівні:

- Адміністратор: має повний доступ до всіх модулів, управління користувачами, налаштування глобальних параметрів системи та інтеграцій.
- Тімлід: може створювати команди, призначати медіабаєрів до креативників та переглядати агреговану статистику по всій групі.
- Медіабаєр: має доступ до аналітики своїх рекламних кабінетів, створює технічні завдання та контролює їх виконання.
- Креативник: бачить список призначених завдань, завантажує готові результати та отримує фідбек від баєрів щодо ефективності своїх відео.

Вимоги до безпеки та інтеграції з API є критичними для захисту інтелектуальної власності та фінансової інформації агенції. Система повинна гарантувати цілісність та конфіденційність даних на всіх етапах обробки. Ключові аспекти безпеки:

- Шифрування трафіку за допомогою протоколу HTTPS для захисту даних під час передачі між клієнтом та сервером.
- Впровадження Middleware для примусової зміни пароля під час першого входу користувача в систему за тимчасовим паролем.
- Валідація та очищення всіх вхідних даних для захисту від поширених вебзагроз, таких як SQL-ін'єкції та XSS-атаки.
- Логування дій користувачів у системі для можливості аудиту змін статусів завдань та фінансових налаштувань.
- Коректна обробка лімітів запитів TikTok API для уникнення блокування доступу під час синхронізації даних.

Чітке формулювання цих вимог дозволяє перетворити абстрактну бізнес-ідею на конкретний план розробки. Дотримання технічних стандартів на етапі специфікації забезпечує масштабованість системи та спрощує процес тестування у майбутньому. Це створює умови для створення надійного інструменту, який не лише вирішує поточні проблеми координації, а й може адаптуватися до нових викликів ринку цифрової реклами.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Обґрунтування вибору технологічного стеку та архітектурного патерну

Проєктування архітектури сучасної аналітичної платформи вимагає вибору таких технологічних рішень, які б забезпечували стабільну роботу при обробці великих масивів даних у режимі реального часу. Основним архітектурним патерном для розробки системи TTMON було обрано Model-View-Template. Цей паттерн є фундаментальним для обраного фреймворку Django [8] і дозволяє досягти високого рівня модульності програмного коду. У межах цієї концепції рівень моделі відповідає за структуру даних та взаємодію з базою, рівень шаблону визначає спосіб представлення інформації користувачеві, а рівень представлення виступає сполучною ланкою, що містить основну бізнес-логіку системи. Коли користувач взаємодіє з клієнтською частиною додатка, його браузер формує відповідний мережевий запит, який спочатку потрапляє до серверного маршрутизатора. Маршрутизатор аналізує адресу та спрямовує запит до відповідного контролера на рівні представлення. Цей контролер перевіряє права доступу згідно з налаштованою рольовою моделлю та за потреби ініціює звернення до рівня моделі. Модель, використовуючи механізми об'єктно-реляційного відображення, взаємодіє з базою даних і повертає необхідні аналітичні масиви. Далі рівень представлення передає цю інформацію до шаблонізатора, який органічно поєднує отримані динамічні дані з візуальною структурою інтерфейсу. Згенерований гіпертекстовий документ повертається назад, дозволяючи системі сформувати фінальну відповідь. Такий строгий розподіл відповідальності дозволяє незалежно оновлювати окремі частини додатка, що є критично важливим при постійній зміні вимог до маркетингових інструментів [8]. Послідовність виконання внутрішніх операцій у межах цього архітектурного підходу, починаючи від отримання вхідного запиту сервером і закінчуючи формуванням фінальної відповіді клієнту, проілюстрована на діаграмі активності.

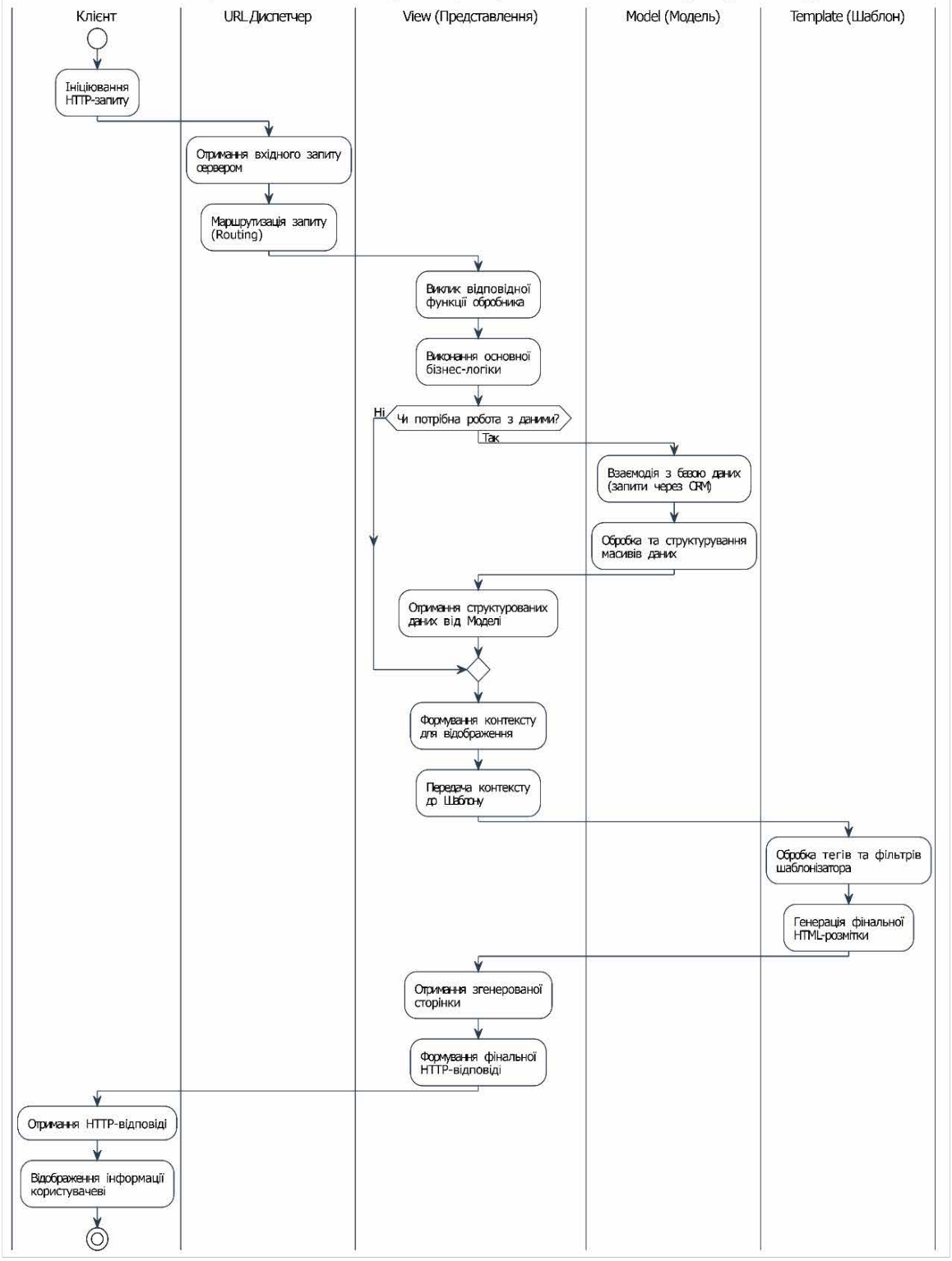

Рис. 2.1. Діаграма активності процесу обробки HTTP-запиту в архітектурі MVT

Вибір мови програмування Python та фреймворку Django обумовлений їхньою здатністю до швидкого масштабування та наявністю потужних засобів безпеки безпосередньо в базовій конфігурації. Django дотримується філософії

включення всіх необхідних інструментів за замовчуванням, що дозволяє зосередитися на розробці унікального функціоналу моніторингу. Використання об'єктно-реляційного відображення у Django значно прискорює роботу з даними, перетворюючи записи таблиць на об'єкти мови Python, що мінімізує ймовірність виникнення синтаксичних помилок у складних SQL-запитах. Крім того, автоматична система адміністративного інтерфейсу фреймворку дозволяє швидко налаштувати робоче середовище для керування контентом на початкових етапах розробки [8, с. 1].

Для надійного збереження фінансових показників та результатів синхронізації з TikTok API було обрано реляційну систему управління базами даних PostgreSQL. Ця СУБД вважається стандартом для проєктів, де пріоритетом є цілісність та точність даних. Завдяки повній відповідності принципам ACID, PostgreSQL гарантує, що жодна фінансова транзакція або аналітичний звіт не будуть втрачені чи спотворені внаслідок технічних збоїв. Особливою перевагою цієї бази даних для проєкту є її ефективна робота з типом даних JSONB, що дозволяє зберігати неструктуровані відповіді від зовнішніх сервісів у компактному двійковому форматі зі збереженням можливості індексації та швидкого пошуку за окремими полями [9].

Крім того, обраний технологічний стек забезпечує високий рівень стійкості системи до раптових пікових навантажень, які неминуче виникають під час масового запуску нових рекламних матеріалів. Архітектура фреймворку дозволяє легко масштабувати обчислювальні потужності шляхом паралельного запуску додаткових робочих процесів, які одночасно обробляють запити від медіабаєрів та креативних дизайнерів. Зі свого боку база даних використовує ефективні механізми внутрішнього кешування планів запитів та інтелектуального управління пулом з'єднань. Це запобігає перевантаженню сервера навіть під час активної фонової синхронізації великих обсягів статистики через програмні інтерфейси соціальної мережі. Усі ці програмні рішення логічно об'єднуються у єдину відмовостійку інфраструктуру, яка слугує надійним фундаментом для безперебійного функціонування процесів агенції. Завдяки впровадженню сучасних інструментів безпеки комунікація між усіма

компонентами надійно захищена від стороннього втручання. Регулярне резервне копіювання даних автоматизовано на рівні серверної операційної системи. Це гарантує максимально швидке відновлення працездатності після можливих системних збоїв. Додаткове балансування мережевого трафіку дозволяє рівномірно розподіляти запити між вузлами. Такий підхід не тільки підвищує загальну продуктивність комплексу, а й суттєво мінімізує ризики простою під час проведення планових профілактичних робіт. Детальне відображення фізичного розташування цих обчислювальних вузлів, серверів та мережевих зв'язків між ними наведено на відповідній візуальній схемі розгортання.

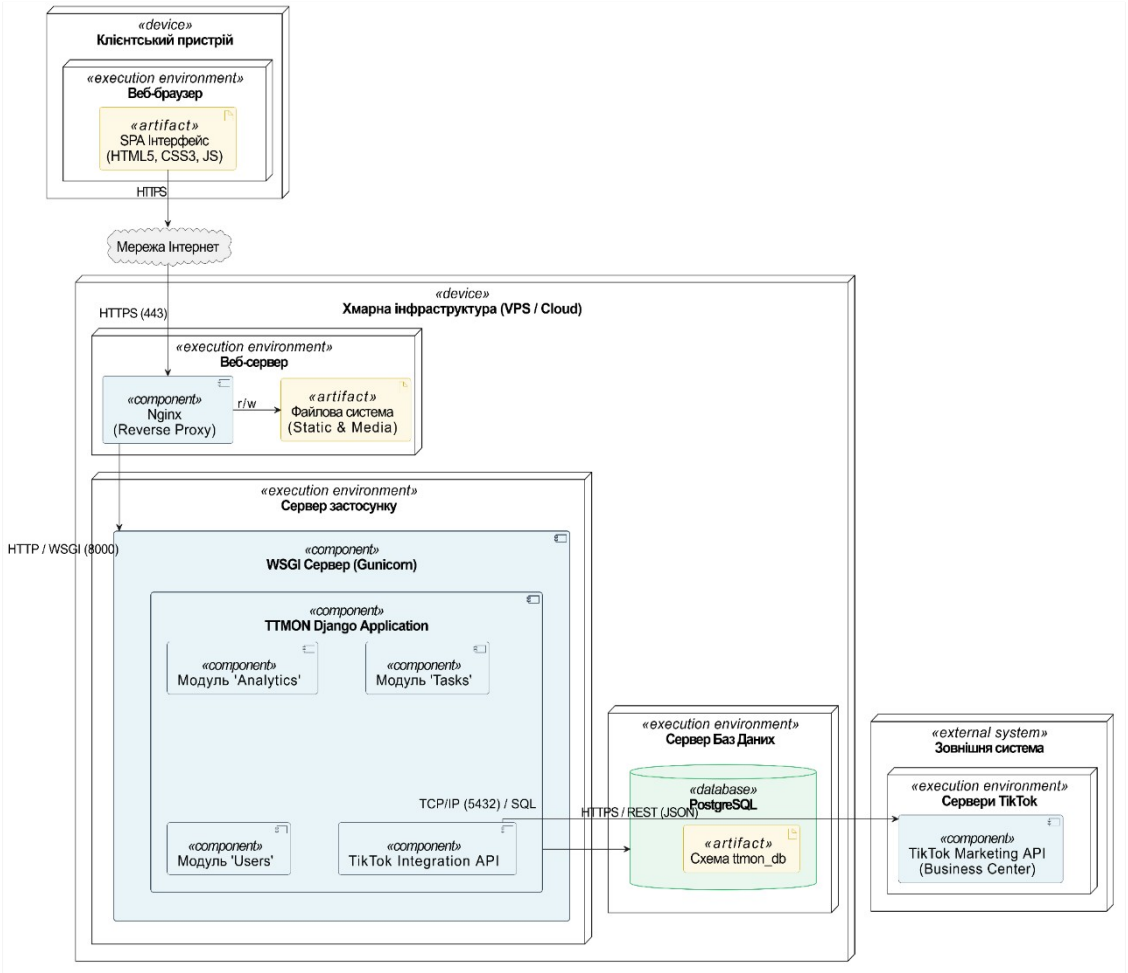

Рис. 2.2. Діаграма розгортання серверної інфраструктури та компонентів системи TTMON

Серверна архітектура додатка передбачає чітке розділення середовища виконання та конфігураційних параметрів. Для забезпечення безпеки секретних даних, таких як ключі доступу до API TikTok та паролі адміністратора,

використовується підхід із застосуванням змінних середовища та файлів формату .env. Це дозволяє зберігати чутливу інформацію окремо від вихідного коду програми, що є обов'язковою вимогою при веденні розробки з використанням систем контролю версій. У разі потрапляння коду у відкритий доступ, конфіденційні дані залишаться захищеними на сервері, а розробники зможуть легко змінювати параметри підключення до бази даних чи налаштування безпеки без необхідності внесення виправлень у логіку роботи самого додатка.

Поєднання Django та PostgreSQL створює стійку екосистему, яка здатна обробляти сотні тисяч записів про рекламні креативи без помітного зниження продуктивності. Використання Python як основної мови розробки відкриває доступ до великої кількості математичних бібліотек, які у майбутньому можуть бути задіяні для прогнозування вигорання креативів на основі історичних даних. Обґрунтованість такого стеку технологій підтверджується не лише їхньою популярністю у професійній спільноті, а й високою швидкістю розробки прототипів, що дозволяє оперативно перевіряти гіпотези щодо ефективності нових інструментів моніторингу. Кожен компонент обраної архітектури працює над досягненням головної мети – створення стабільного, безпечного та прозорого інструменту для автоматизації маркетингових процесів.

Впровадження архітектурного паттерна MVT у поєднанні з реляційною базою даних PostgreSQL забезпечує системі TTMON необхідну гнучкість для подальшого розширення функціоналу. Завдяки вбудованим механізмам міграцій у Django зміна структури бази даних при додаванні нових метрик відбувається плавно та без ризику пошкодження наявних записів. Такий підхід гарантує тривалий життєвий цикл програмного продукту та можливість його адаптації під будь-які зміни в алгоритмах рекламних платформ. Обрана стратегія проєктування дозволяє створити продукт, який є водночас технічно складним всередині та простим у налаштуванні і підтримці для системних адміністраторів агенції.

2.2. Проєктування концептуальної та логічної моделі бази даних

Основою для стабільної та безперебійної роботи будь-якої інформаційної системи є правильно спроєктована архітектура бази даних. У контексті розробки системи моніторингу рекламної ефективності TTMON головним завданням етапу проєктування стало створення такої реляційної моделі, яка б змогла органічно об'єднати два паралельні потоки даних: строгу фінансову аналітику з рекламних кабінетів та гнучкий процес управління творчими завданнями. Концептуальна модель бази даних будується на основі виділення ключових сутностей предметної області, визначення їхніх атрибутів та встановлення логічних зв'язків між ними, що дозволяє уникнути дублювання інформації та забезпечити каскадне оновлення або видалення пов'язаних записів.

Фундаментом розробленої логічної моделі виступає підсистема управління користувачами та їхніми ролями. Головною таблицею тут є users_customuser, яка розширює стандартну модель користувача фреймворку та містить розширений набір атрибутів для ідентифікації співробітників агенції. Ця таблиця зберігає базові ідентифікаційні дані, такі як електронна пошта, ім'я та хешований пароль, а також набір специфічних булевих прапорців, які відповідають за контроль першого входу та управління доступом. Кожен користувач має визначену роль у системі, що фіксується у відповідному текстовому полі, і може бути прив'язаний до конкретної команди через зовнішній ключ, який вказує на сутність users_team. Таблиця команд містить інформацію про назву робочої групи та посилання на лідера команди, що дозволяє вибудовувати ієрархічну структуру підпорядкування всередині маркетингової агенції.

Для забезпечення взаємодії із зовнішніми сервісами соціальної мережі логічна модель передбачає наявність сутностей інтеграції. Таблиця users_tiktokintegration відповідає за безпечне зберігання токенів доступу та ідентифікаторів профілів, отриманих за протоколом авторизації. Ця сутність пов'язана з таблицею analytics_tiktokadaccount, яка зберігає конфігураційні параметри конкретних рекламних кабінетів, включаючи валюту білінгу, часовий

пояс та поточний статус активності. Зв'язок між кабінетами та безпосередніми виконавцями реалізується через проміжну таблицю дозволів, що дозволяє гнучко надавати доступ різним медіабаєрам до аналітики одного клієнтського акаунта без необхідності повторної авторизації через соціальну мережу.

Найбільш критичним сегментом бази даних є блок фінансової аналітики та моніторингу креативів. Сутність analytics_campaign виступає агрегатором даних для кожної окремої рекламної кампанії. Вона містить вичерпний набір метрик, серед яких загальний бюджет, фактичні витрати, кількість показів, кліків та здійснених конверсій. Крім числових показників, таблиця зберігає інформацію про географію націлювання, дати початку та завершення активності. Ця сутність має зв'язок типу один-до-багатьох із таблицею analytics_adcreative, яка зберігає статистику щодо кожного окремого рекламного відеоролика всередині кампанії. Саме в цій таблиці реалізовано ключову бізнес-логіку системи моніторингу: окрім фінансових витрат та розрахованого показника ROI, кожен запис містить булеве поле is_burning_out. Зміна статусу цього поля на основі математичних алгоритмів сигналізує системі про те, що конкретний креатив втратив свою ефективність і потребує негайної заміни.

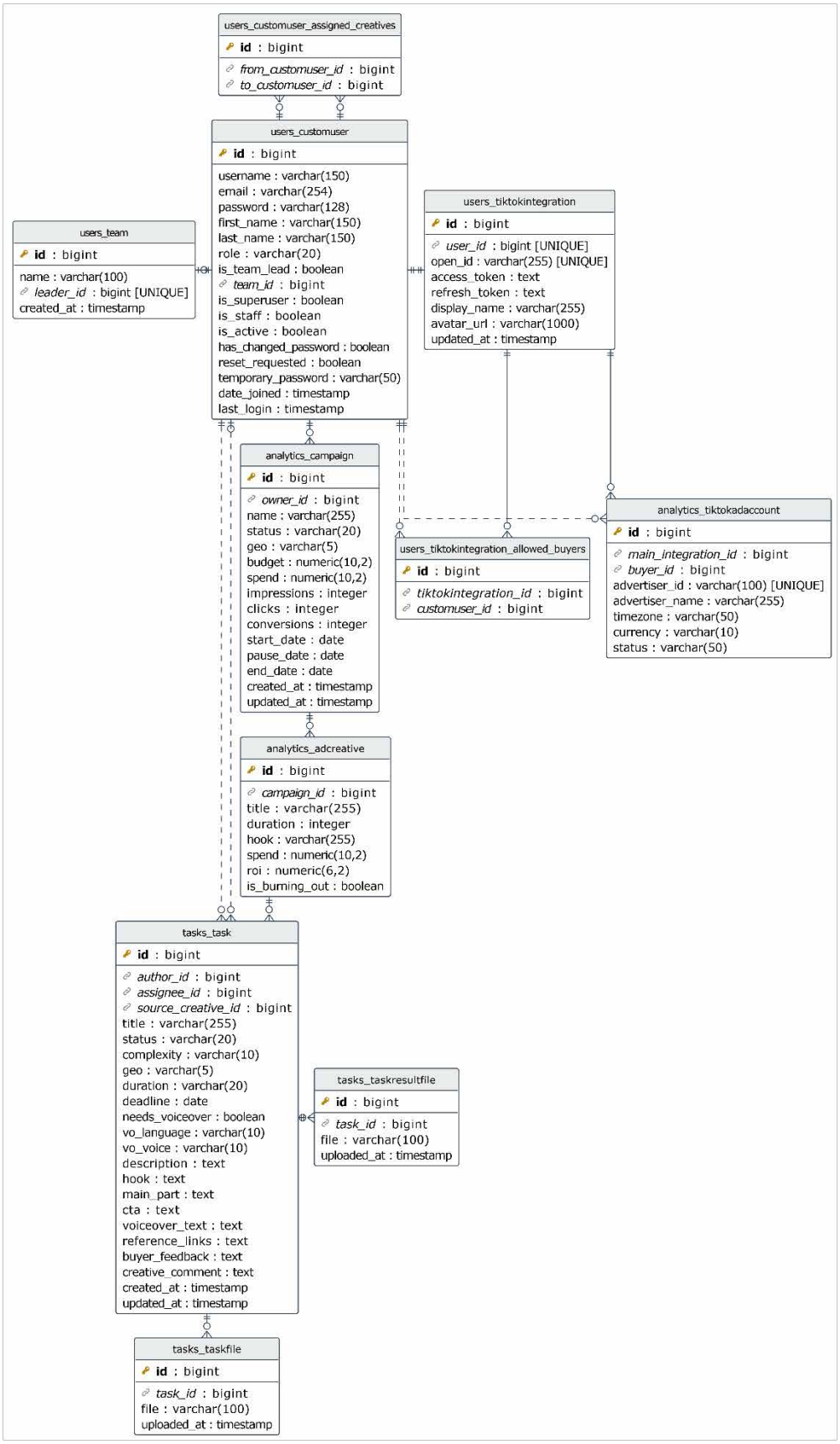

Рис. 2.3. ER-діаграма концептуальної та логічної моделі бази даних системи TTMON

Виявлення факту вигорання відеоролика запускає процес створення технічного завдання на виробництво нового контенту, що відображається у зв'язках із блоком управління завданнями. Центральною сутністю цього блоку є таблиця tasks_task, яка виступає мостом між аналітикою та креативним відділом. Кожне завдання створюється конкретним медіабаєром і призначається відповідному дизайнеру через ключі author_id та assignee_id. Найважливішим архітектурним рішенням є наявність у цій таблиці поля source_creative_id, яке встановлює прямий зв'язок із сутністю analytics_adcreative. Це означає, що кожне нове завдання не існує у вакуумі – воно завжди прив'язане до конкретного вигорілого відео, що дозволяє дизайнеру миттєво зрозуміти контекст проблеми.

Структура таблиці tasks_task максимально деталізована для потреб виробництва контенту в мережі TikTok. Вона містить специфічні текстові поля для опису різних етапів відеоролика: візуального гачка уваги, основної змістовної частини та заклику до дії. Окрім цього, модель передбачає зберігання параметрів озвучення, де вказується необхідність використання голосу диктора, мови, емоційного забарвлення та безпосередньо самого тексту для озвучення. Життєвий цикл завдання контролюється через поле статусу, а комунікація між відділами фіксується у полях зворотного зв'язку замовника та коментарях креативника. Така структура таблиці дозволяє повністю відмовитися від використання сторонніх месенджерів для передачі вимог.

Для зберігання медіафайлів логічна модель використовує дві розділені сутності: tasks_taskfile та tasks_taskresultfile. Такий поділ є цілеспрямованим рішенням, яке гарантує чітке розмежування вхідних та вихідних матеріалів. У першу таблицю медіабаєри завантажують референсні відео, шрифти, логотипи або інші матеріали, необхідні для початку роботи. У другу таблицю креативники завантажують фінальні відрендерені відеоролики, які передаються на перевірку. Обидві таблиці пов'язані із сутністю завдання зв'язком багато-до-одного, що дозволяє прикріплювати необмежену кількість файлів до одного технічного завдання. Наявність міток часу для кожного завантаженого файлу дає змогу керівництву відстежувати реальні терміни виконання робіт.

Побудована ER-діаграма демонструє високий рівень нормалізації даних, що унеможливлює виникнення аномалій оновлення або видалення інформації. Використання цілісної системи зовнішніх ключів гарантує референційну цілісність: наприклад, система не дозволить видалити рекламну кампанію, якщо до неї прив'язані активні креативи, а видалення користувача не призведе до втрати створених ним технічних завдань завдяки правильному налаштуванню каскадних операцій. Запропонована структура бази даних ідеально відповідає обраному архітектурному патерну та закладає надійний фундамент для програмної реалізації всіх функціональних вимог системи моніторингу.

2.3 Розробка моделі рольового доступу RBAC та командної структури

Ефективне управління маркетинговою агенцією неможливе без чіткого розмежування зон відповідальності між співробітниками та суворого контролю доступу до конфіденційної комерційної інформації. У процесі проєктування системи TTMON особливу увагу було приділено розробці гнучкої та безпечної моделі управління доступом на основі ролей, яка у міжнародній практиці позначається абревіатурою RBAC. Цей архітектурний підхід передбачає надання користувачам прав доступу до інформаційних ресурсів виключно на основі їхніх посадових обов'язків та місця в організаційній структурі компанії. Використання вбудованих механізмів авторизації фреймворку дозволяє реалізувати цю парадигму на базовому рівні системи, забезпечуючи перевірку дозволів при кожному зверненні до серверних маршрутів або спробі зміни записів у базі даних [10]. Такий підхід гарантує, що фінансова статистика клієнтів агенції буде надійно захищена від несанкціонованого перегляду або випадкового пошкодження некомпетентними користувачами.

Фундаментом розробленої моделі є виділення чотирьох ключових ролей, кожна з яких наділена специфічним набором функціональних можливостей. Найвищий рівень привілеїв належить системному адміністратору, який відповідає за глобальне налаштування платформи. Ця роль має ексклюзивне право додавати нових співробітників, формувати робочі команди, деактивувати

облікові записи звільнених працівників та налаштовувати критично важливі параметри інтеграції з програмними інтерфейсами соціальних мереж. Адміністратор бачить усю картину роботи агенції, проте його головна функція зводиться до технічного забезпечення процесів, а не до безпосередньої участі у створенні реклами. Наступним щаблем в ієрархії є керівник команди або тімлід, який здійснює операційне управління групою фахівців. Тімлід отримує доступ до агрегованої аналітики всіх рекламних кампаній своєї команди, що дозволяє йому відстежувати загальну рентабельність проєктів, перерозподіляти бюджети та контролювати рівень завантаженості підлеглих.

Безпосередніми виконавцями маркетингових завдань виступають медіабаєри та креативні дизайнери, взаємодія яких формує основний потік даних у системі. Роль медіабаєра передбачає наявність доступу до фінансових дашбордів виключно тих рекламних кабінетів, які були йому призначені керівництвом. Баєр безперервно аналізує метрики ефективності, виявляє ознаки вигорання відеороликів і на основі цих даних формує технічні завдання на створення нового контенту. Своєю чергою, роль дизайнера є найбільш ізольованою з точки зору доступу до комерційної таємниці. Креативник взагалі не бачить фінансової статистики, вартості залучення клієнтів чи загальних бюджетів кампаній. Його робочий простір обмежується виключно дошкою завдань, де він може переглядати призначені йому доручення, завантажувати готові медіафайли та читати коментарі від замовників. Детальний розподіл цих прав та специфіку взаємодії кожної ролі з основними модулями платформи наочно зображено на відповідній схемі прецедентів.

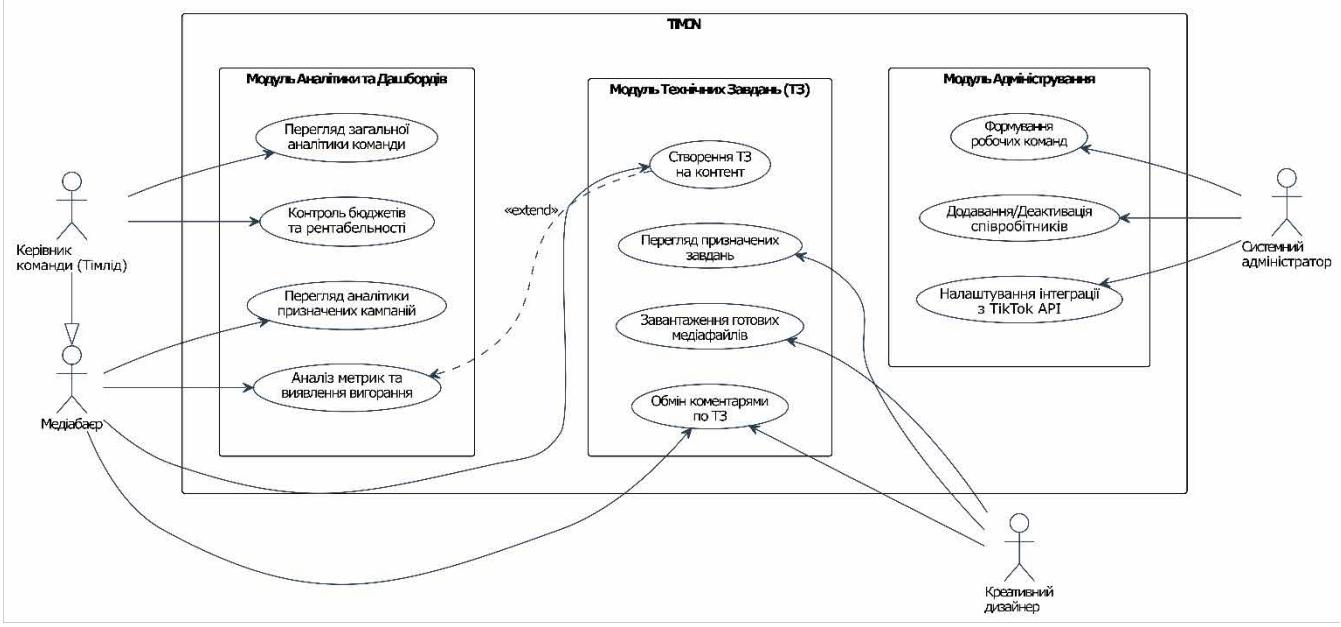

Рис. 2.4 - Діаграма прецедентів взаємодії користувачів із підсистемами платформи

Організаційна структура сучасної маркетингової агенції рідко буває лінійною, що вимагає впровадження складної логіки зв'язків між співробітниками на рівні бази даних. У класичному виробничому циклі один медіабаєр може одночасно вести кілька десятків кампаній для різних ніш, кожна з яких потребує специфічного візуального оформлення. Для створення анімаційних роликів йому потрібен один фахівець, а для монтажу живого відео – інший. З іншого боку, талановитий креативник рідко працює лише з одним баєром – він отримує завдання від різних фахівців із закупівлі реклами в межах своєї команди. Така специфіка роботи зумовлює необхідність реалізації логіки зв'язку багато-до-багатьох між сутностями баєрів та дизайнерів. Цей архітектурний підхід дозволяє керівнику команди гнучко формувати робочі зв'язки, підключаючи необхідних фахівців до конкретних завдань без жорсткої прив'язки одного працівника до іншого. Система автоматично фільтрує списки доступних виконавців під час створення технічного завдання, гарантуючи, що баєр зможе призначити роботу лише тому дизайнеру, який офіційно входить до складу його робочої групи.

Ключовим артефактом, навколо якого будується взаємодія між цими двома ролями, є технічне завдання на створення рекламного креативу. Оскільки процес розробки відеоролика є багатоетапним і вимагає постійного узгодження, система повинна мати механізм суворого контролю життєвого циклу цього документа. Для вирішення цієї проблеми було спроєктовано логіку на основі кінцевого автомата, де кожне завдання може перебувати лише в одному з чітко визначених станів у конкретний момент часу. Перехід між цими станами відбувається виключно за умови виконання певних системних тригерів або дій користувачів, що повністю виключає можливість хаотичної зміни статусів або втрати актуальної інформації про етап виконання робіт.

Життєвий цикл технічного завдання розпочинається зі стану чорновика. На цьому етапі медіабаєр збирає необхідні референси, формулює вимоги до тексту та візуальних ефектів, але завдання ще не видиме для виконавця. Після завершення оформлення баєр публікує документ, і система автоматично переводить його у стан виконання, надсилаючи відповідне сповіщення призначеному дизайнеру. Отримавши завдання, креативник бере його в роботу, і з цього моменту запускається таймер дедлайну. Коли відео змонтоване та завантажене на сервер, дизайнер натискає кнопку завершення, що змінює стан завдання на перевірку. Цей статус блокує можливість внесення змін до медіафайлів зі сторони виконавця та передає ініціативу назад до медіабаєра. Усі ці переходи та можливі сценарії розгалуження детально задокументовані за допомогою візуальної моделі станів.

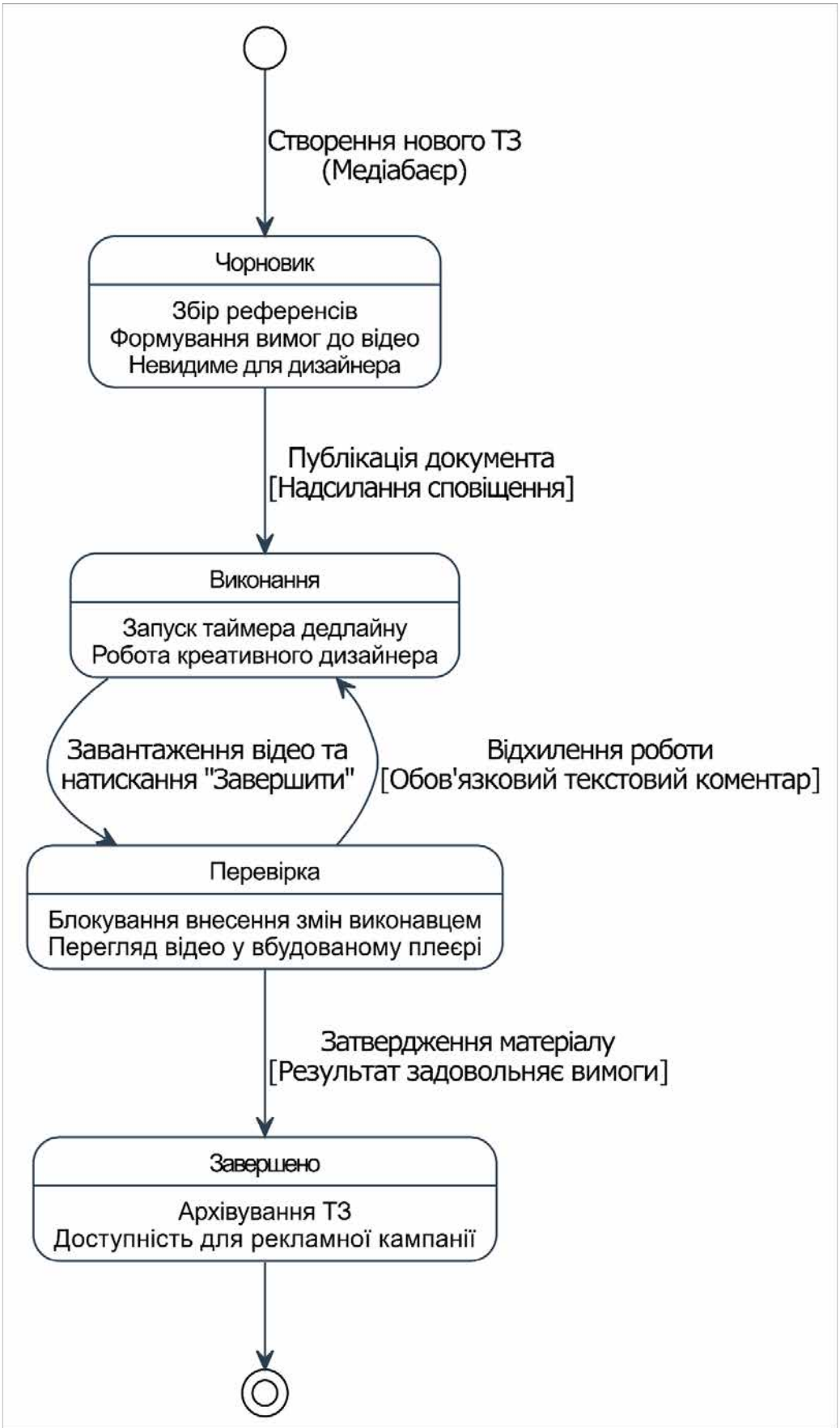

Рис. 2.5 - Діаграма станів життєвого циклу технічного завдання

Етап перевірки є критичним для забезпечення високої якості кінцевого продукту. Медіабаєр переглядає завантажений відеоролик безпосередньо у вбудованому плеєрі системи та приймає рішення щодо його відповідності початковим вимогам. Якщо результат повністю задовольняє замовника, він затверджує матеріал, і завдання переходить у фінальний стан завершеності. З цього моменту воно архівується та стає доступним для прикріплення до конкретної рекламної кампанії в модулі аналітики. Однак у випадку виявлення помилок або невідповідностей баєр має можливість відхилити роботу, обов'язково вказавши причину в текстовому коментарі. У такій ситуації система повертає завдання на етап виконання, і дизайнер отримує змогу завантажити оновлену версію файлу. Цей ітеративний процес може повторюватися кілька разів до повного затвердження, при цьому система зберігає всю історію змін файлів та коментарів для можливого аудиту керівником команди у майбутньому.

Впровадження такої строгої моделі станів у поєднанні з рольовим доступом кардинально змінює культуру комунікації всередині маркетингової агенції. Відмова від розрізнених месенджерів на користь єдиного стандартизованого процесу знімає психологічне напруження між співробітниками, оскільки всі вимоги фіксуються документально, а причини відхилення робіт є прозорими та обґрунтованими. Крім того, наявність чітких статусів дозволяє тімліду генерувати автоматичні звіти про середню швидкість виконання завдань кожним дизайнером та відстежувати відсоток робіт, які повертаються на доопрацювання. У сукупності ці технічні рішення перетворюють програмний продукт з простого інструменту перегляду статистики на повноцінну платформу управління операційною діяльністю підприємства, де кожен співробітник працює у комфортному інформаційному полі, зосереджуючись виключно на своїх професійних компетенціях.

2.4 Проєктування процесу інтеграції з TikTok Marketing API

Сучасні системи моніторингу рекламної ефективності не можуть покладатися на ручне експортування та введення статистичних даних, оскільки

це неминуче призводить до затримок, критичних людських помилок та втрати актуальності інформації. Для платформи TTMON ключовим архітектурним рішенням є розробка надійного модуля автоматичної та безперервної синхронізації з офіційними серверами соціальної мережі через TikTok Marketing API. Цей захищений програмний інтерфейс надає сертифікованим розробникам стандартизований доступ до всіх фінансових та статистичних показників рекламних кабінетів агенції у режимі реального часу. Проєктування процесу інтеграції вимагає комплексного вирішення двох фундаментальних інженерних завдань: забезпечення безпечної процедури авторизації користувачів для отримання доступу до їхніх комерційних даних та побудови відмовостійкого фонового конвеєра для отримання, математичної обробки й аналізу великих масивів інформації.

Безпека доступу до клієнтських рекламних акаунтів гарантується використанням відкритого протоколу авторизації OAuth 2.0, який є загальноприйнятим світовим галузевим стандартом для делегування прав доступу без необхідності прямої передачі паролів [12]. Процес інтеграції розпочинається з ініціалізації запиту медіабаєром або системним адміністратором у клієнтському інтерфейсі платформи TTMON. На цьому етапі користувач перенаправляється на офіційну захищену сторінку авторизації TikTok, де він повинен пройти автентифікацію та надати явну згоду на читання аналітичних даних своїм рекламним кабінетом. Після успішного підтвердження дозволів сервери соціальної мережі автоматично повертають користувача назад на заздалегідь вказану адресу перенаправлення нашого бекенду, обов'язково додаючи до запиту спеціальний криптографічний тимчасовий код авторизації.

Отримавши цей код, серверна частина системи TTMON виконує прихований безпечний запит безпосередньо до API платформи, обмінюючи тимчасовий параметр на довгостроковий маркер доступу та маркер оновлення [11]. Отримані цифрові ключі надійно шифруються за допомогою алгоритмів симетричного шифрування та зберігаються у реляційній базі даних PostgreSQL, що дозволяє системі у майбутньому виконувати автоматизовані фонові запити від імені конкретного користувача без його безпосередньої фізичної участі.

Важливо зазначити, що отриманий маркер доступу має жорстко обмежений термін дії, що вимагає реалізації додаткового механізму для його регулярного оновлення. Система автоматично відстежує час експірації токена і використовує маркер оновлення для запиту нових ключів, гарантуючи безперервність роботи аналітичних модулів. Послідовність виконання кожного кроку цього складного криптографічного обміну даними між вебклієнтом, нашим сервером та серверами авторизації соціальної мережі детально відображена на діаграмі послідовності.

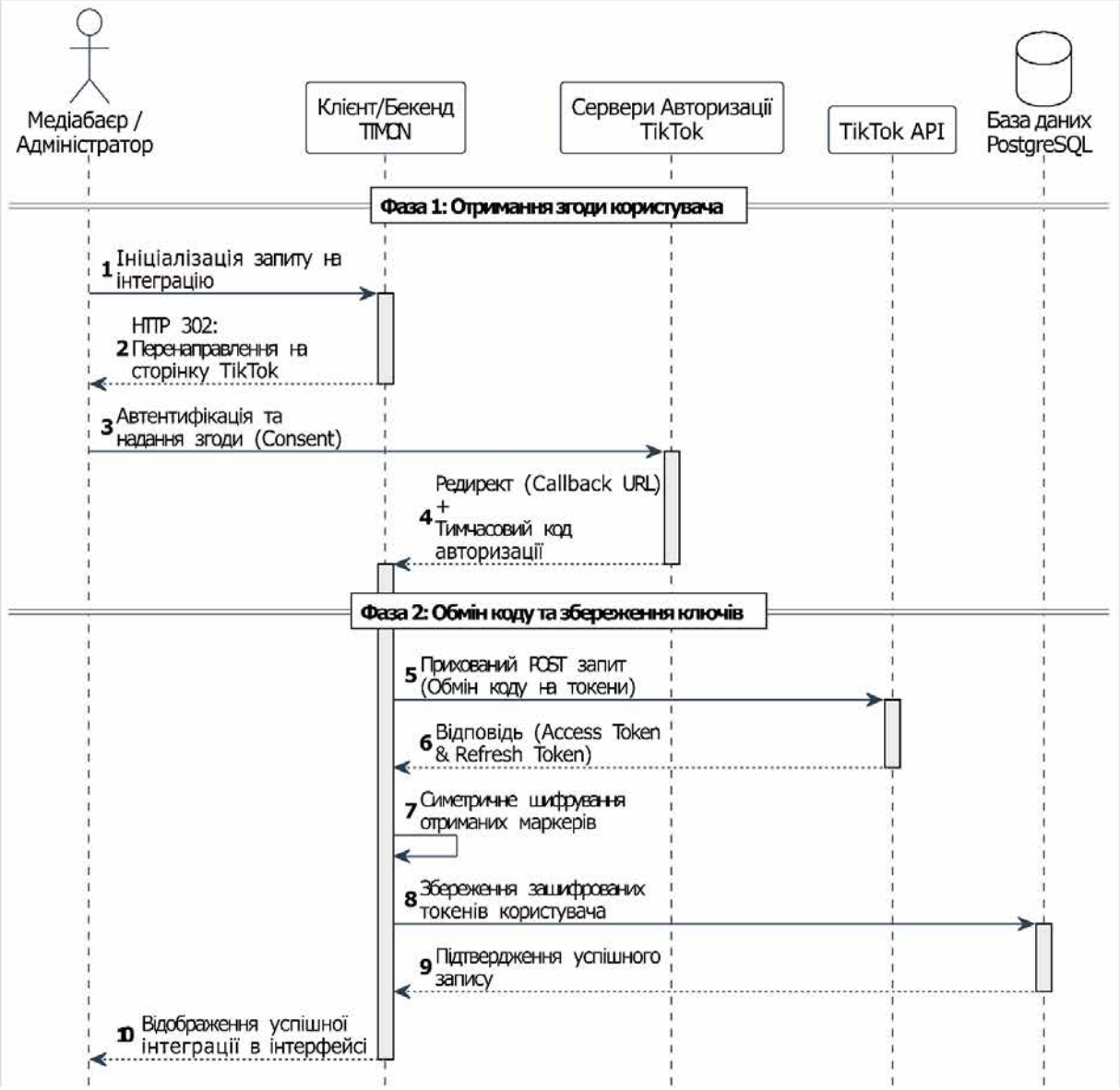

Рис. 2.6 – Діаграма послідовності процесу авторизації OAuth 2.0

Після успішного налаштування захищеного каналу зв'язку та закріплення необхідних дозволів система переходить до етапу регулярної агрегації

статистичних даних. Оскільки рекламні кампанії агенції генерують величезний потік інформації щосекунди, прямі синхронні запити під час відкриття користувачем сторінки дашборду є вкрай неефективними і можуть призвести до критичного блокування інтерфейсу. З огляду на це, архітектура передбачає використання асинхронних фонових завдань, які запускаються планувальником за чітко визначеним розкладом. Сервер формує пакетні HTTP-запити до відповідних кінцевих точок TikTok API для отримання вичерпних звітів на рівні кожного окремого креативу. Отриманий масив даних у форматі JSON проходить обов'язковий етап десеріалізації, структурної валідації та нормалізації, після чого система переходить до виконання закладеної бізнес-логіки. Для кожного рекламного відеоролика внутрішній алгоритм розраховує актуальні показники коефіцієнта клікабельності та загальної рентабельності інвестицій на основі прямого співвідношення витраченого фінансового бюджету до кількості реально отриманих конверсій.

Розраховані фінансові метрики не просто зберігаються у таблицях бази даних для їхньої подальшої візуалізації, а обов'язково проходять через спеціалізований модуль аналітичного контролю. Саме цей модуль відповідає за своєчасне виявлення перших ознак вигорання рекламного контенту. Алгоритм порівнює поточну короткострокову динаміку зміни рентабельності інвестицій із ретроспективними даними цього ж креативу за попередні успішні періоди, а також з еталонними пороговими значеннями, які були попередньо встановлені тімлідом для конкретної кампанії. Якщо система чітко фіксує стійке падіння показників нижче критичної межі протягом визначеного проміжку часу, математична модель безапеляційно класифікує такий креатив як вигорілий. Ця подія автоматично ініціює створення відповідного запису в журналі бази даних та миттєву генерацію термінового сповіщення для відповідального фахівця із закупівлі реклами.

Сформоване сповіщення містить пряме посилання на аналітичний дашборд проблемного креативу, що дозволяє медіабаєру оперативно оцінити масштаби фінансових втрат, зупинити неефективну рекламу та одним кліком ініціювати

процедуру постановки нового технічного завдання для креативного відділу дизайнерів. Цей комплексний багатокроковий конвеєр обробки інформації, починаючи від ініціації мережевого запиту до зовнішнього API і закінчуючи генерацією кінцевого попереджувального сповіщення для співробітника агенції, наочно та структурно проілюстровано за допомогою загальноприйнятої нотації моделювання бізнес-процесів.(Рис. 2.7)

Важливим аспектом проєктування надійної інтеграції із зовнішніми програмними інтерфейсами є забезпечення високого рівня стійкості нашої системи до можливих короткочасних мережевих збоїв та суворих обмежень частоти запитів, які встановлюються серверами TikTok для захисту їхньої інфраструктури від перевантажень [11]. У разі перевищення дозволеного ліміту звернень API обов'язково повертає специфічний код помилки, що вимагає від архітектури TTMON здатності тимчасово та безпечно призупинити синхронізацію даних. Для управління такими нестандартними ситуаціями було спроєктовано та впроваджено механізм експоненційної затримки, який автоматично утримує потоки та повторює відхилені запити через прогресивно зростаючі інтервали часу. Це гарантує, що процес агрегації життєво необхідних даних відновиться одразу після зняття обмежень з боку соціальної мережі без жодної втрати важливої статистичної інформації. Такий комплексний, безпечний та відмовостійкий підхід до проєктування інтеграційного модуля забезпечує маркетингову агенцію надійним програмним інструментом, здатним стабільно працювати з сотнями клієнтських рекламних кабінетів одночасно.

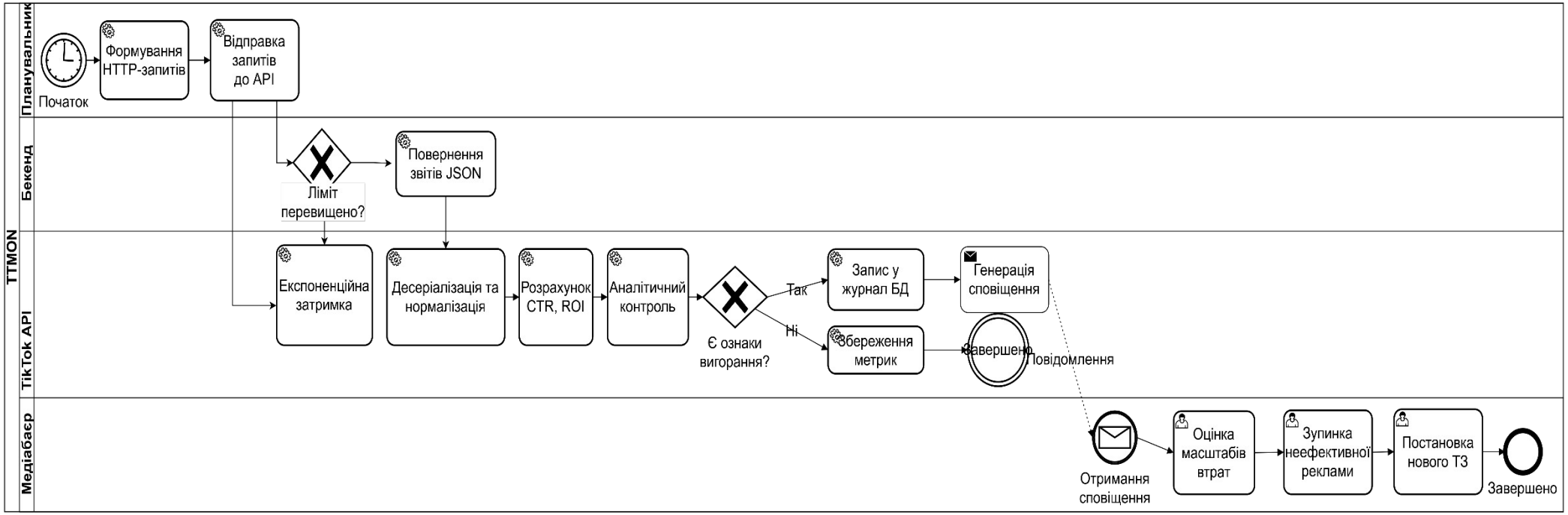

Рис. 2.7 – Діаграма BPMN процесу моніторингу метрик та виявлення вигорання креативу.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛІВ СИСТЕМИ

3.1 Налаштування середовища розробки та базова конфігурація

Процес програмної реалізації інформаційної системи TTMON розпочався з ініціалізації базового середовища розробки. Для написання програмного коду було обрано інтегроване середовище розробки, яке надає розширені інструменти для роботи з фреймворком Django та мовою Python. На початковому етапі було створено віртуальне середовище, що дозволило ізолювати залежності проєкту від глобальних пакетів операційної системи. Після встановлення ядра Django було згенеровано базову структуру каталогів [13, 17с.].

Відповідно до принципів модульності, логіку системи було розділено на три основні додатки: users (управління користувачами, ролями та авторизацією), analytics (інтеграція з API та обробка статистичних даних) та tasks (управління технічними завданнями та медіафайлами). Окрім модулів із бізнес-логікою, структура містить директорії media для збереження завантажених користувачами файлів, static для каскадних таблиць стилів і скриптів, а також templates для HTML-шаблонів. Візуальне відображення архітектури каталогів наведено на рисунку 3.1.

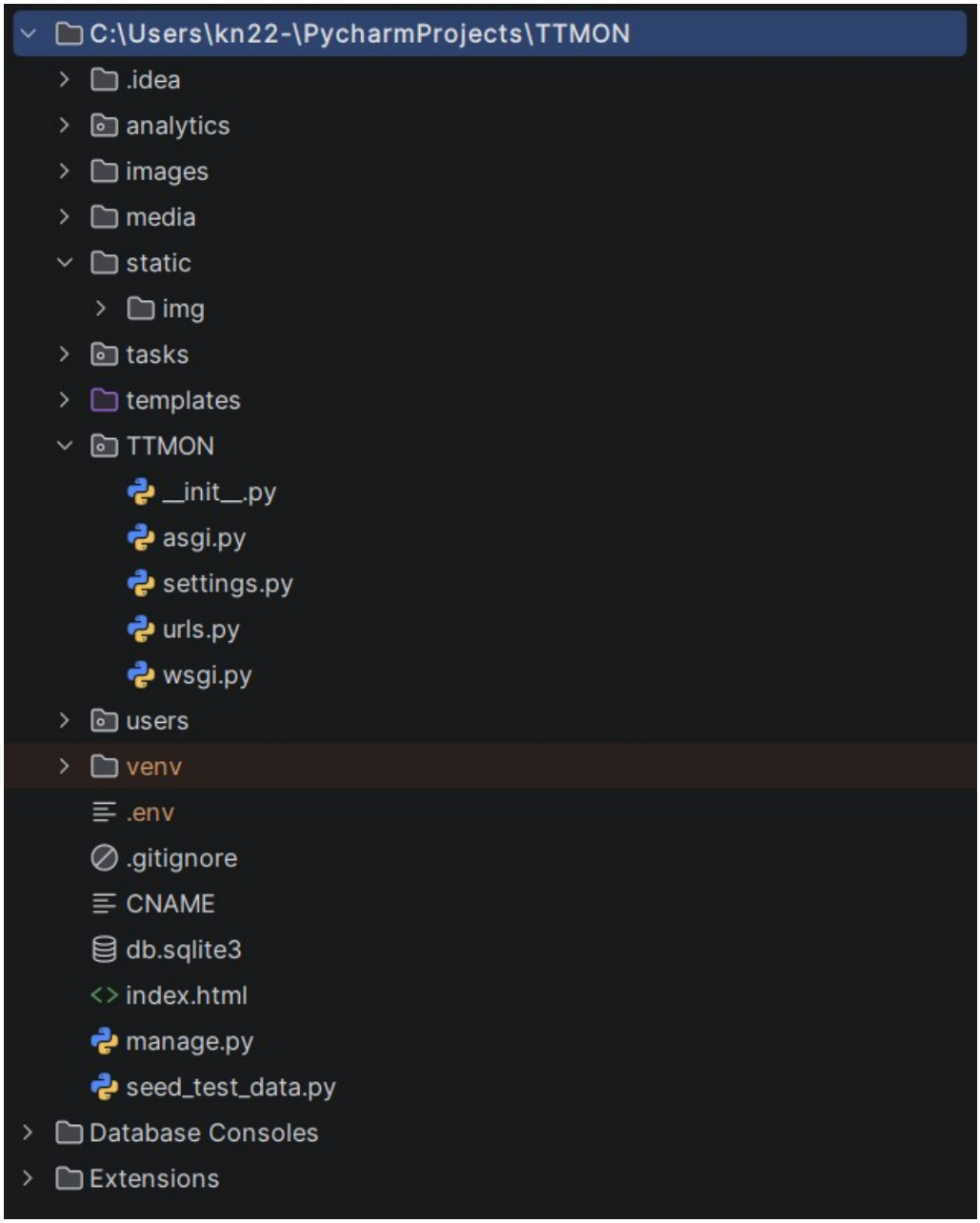

Рис. 3.1 - Структура каталогів та файлів проєкту TTMON у середовищі розробки

Наступним критичним кроком стало налаштування безпеки та управління конфігураційними змінними. Згідно з найкращими практиками веб-розробки, зберігання секретних ключів, паролів до бази даних та токенів доступу до зовнішніх API у відкритому коді є неприпустимим, оскільки це створює критичну вразливість системи [14, 46с.]. Для вирішення цієї проблеми було застосовано бібліотеку python-dotenv. Вона дозволяє завантажувати конфіденційні дані з прихованого файлу .env безпосередньо у змінні середовища

під час запуску сервера. У файлі settings.py цей механізм реалізовано наступним чином (лістинг 3.1).

Лістинг 3.1 Завантаження конфігураційних змінних із файлу .env

```
import os from pathlib
import Path from dotenv
import load_dotenv
BASE_DIR = Path(file).resolve().parent.parent load_dotenv()
SECRET_KEY = os.getenv('DJANGO_SECRET_KEY')
TIKTOK_CLIENT_KEY = os.getenv('TIKTOK_CLIENT_KEY')
TIKTOK_CLIENT_SECRET = os.getenv('TIKTOK_CLIENT_SECRET')
```

Після налаштування безпеки було здійснено перехід від стандартної бази даних SQLite, яка постачається з Django за замовчуванням, до повноцінної реляційної СУБД PostgreSQL. Це рішення обґрунтоване необхідністю обробки великих масивів аналітичних даних та підтримки типу JSONB для збереження відповідей від TikTok Marketing API. Конфігурація підключення у головному файлі налаштувань автоматично зчитує пароль із захищеного середовища (лістинг 3.2).

Лістинг 3.2 Конфігурація підключення до бази даних PostgreSQL

```
DATABASES = {
    'default': { 'ENGINE': 'django.db.backends.postgresql',
    'NAME': 'ttmon_db',
    'USER': 'postgres',
    'PASSWORD': os.getenv('DB_PASSWORD'),
    'HOST': 'localhost', 'PORT': '5432', } }
```

Для того, щоб ядро фреймворку розпізнало створені модулі та застосувало необхідні налаштування безпеки, їх було зареєстровано у відповідних конфігураційних масивах. У список INSTALLED_APPS було додано локальні додатки users, analytics та tasks. Одночасно з цим базову модель користувача було перевизначено на власну CustomUser за допомогою константи AUTH_USER_MODEL, що є обов'язковою вимогою для подальшої реалізації гнучкої системи рольового доступу.

Окрім цього, для забезпечення додаткового рівня безпеки під час першого входу нових співробітників у систему, до списку MIDDLEWARE було інтегровано власне проміжне програмне забезпечення ForcePasswordChangeMiddleware. Цей компонент перехоплює всі HTTP-запити та перевіряє статус актуальності пароля користувача до того, як запит досягне основної бізнес-логіки (лістинг 3.3).

Лістинг 3.3 Реєстрація додатків та проміжного програмного забезпечення

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
     'django.contrib.staticfiles',
     'users', 'analytics', 'tasks',
    ]
AUTH_USER_MODEL = 'users.CustomUser'
MIDDLEWARE = [
                                    'django.middleware.security.SecurityMiddleware',
'django.contrib.sessions.middleware.SessionMiddleware',
'django.middleware.common.CommonMiddleware',
'django.middleware.csrf.CsrfViewMiddleware',
'django.contrib.auth.middleware.AuthenticationMiddleware',
'django.contrib.messages.middleware.MessageMiddleware',
'django.middleware.clickjacking.XFrameOptionsMiddleware',
'users.middleware.ForcePasswordChangeMiddleware',
]
```

Таким чином, базове налаштування середовища розробки сформувало надійний фундамент для подальшої програмної реалізації системи. Створена архітектура відповідає стандартам безпеки, забезпечує чітку модульність

компонентів та гарантує готовність інфраструктури до роботи з реляційними базами даних.

3.2 Реалізація підсистеми автентифікації та примусової зміни пароля

Розширення стандартного механізму автентифікації фреймворку є критично важливою процедурою для систем із розподіленою рольовою моделлю доступу. Стандартна модель користувача Django надає лише базові поля, такі як логін, електронна пошта та пароль, що є недостатніми для реалізації логіки управління маркетинговою агенцією. З огляду на це, у додатку users було створено власну модель CustomUser, яка успадковує властивості класу AbstractUser [13, 53 с.].

Головною перевагою такого підходу є збереження всіх вбудованих інструментів безпеки, хешування паролів та механізмів сесій із одночасним додаванням кастомних атрибутів. У модель було інтегровано поле визначення ролі користувача role із чітким переліком доступних варіантів, а також булевий прапорець is_first_login, який за замовчуванням має значення True і слугує тригером для примусового оновлення облікових даних. Програмна реалізація цієї моделі представлена у файлі models.py додатку users (лістинг 3.4).

Лістинг 3.4 Програмний код моделі користувача CustomUser

```
from django.contrib.auth.models
import AbstractUser from django.db
import models
class CustomUser(AbstractUser):
ROLE_CHOICES = [ ('admin', 'Administrator'), ('team_lead', 'Team Lead'), ('buyer', 'Media Buyer'), ('creator', 'Creator'), ]
role = models.CharField(max_length=20,
choices=ROLE_CHOICES,
default='creator') is_first_login = models.BooleanField(default=True)
```

Перевизначення моделі на початковому етапі розробки дозволило уникнути складних конфліктів під час генерації первинних міграцій бази даних і заклало основу для проєктування взаємозв'язків між іншими сутностями платформи [14,

235с.]. Для автоматизації контролю безпеки та захисту облікових записів від несанкціонованого використання адміністратором системи було розроблено кастомне проміжне програмне забезпечення middleware. Його призначення полягає в перехопленні кожного вхідного HTTP-запиту від клієнта та перевірці поточного стану користувача.

Логіка роботи ForcePasswordChangeMiddleware базується на аналізі двох ключових умов. Якщо користувач є успішно автентифікованим у системі, але прапорець першого входу is_first_login досі активований, компонент блокує доступ до будь-яких аналітичних чи виробничих сторінок сайту. Спеціальний масив allowed_urls визначає виключення з цього правила, до яких належать безпосередньо сторінка зміни пароля та маршрут завершення сесії, щоб уникнути зациклення перенаправлень. Усі інші запити примусово перенаправляються на форму оновлення пароля. Програмна структура цього компонента реалізована у файлі middleware.py (лістинг 3.5).

Лістинг 3.5 Проміжне програмне забезпечення ForcePasswordChangeMiddleware

```
from django.shortcuts
import redirect from django.urls
import reverse
class ForcePasswordChangeMiddleware:
    def init(self, get_response): self.get_response = get_response
    def __call__(self, request):
        if request.user.is_authenticated and request.user.is_first_login:
            allowed_urls = [reverse('password_change'), reverse('logout')]
            if request.path not in allowed_urls and not request.path.startswith('/static/'):
                return redirect('password_change')
        return self.get_response(request)
```

Впровадження цього Middleware дозволяє повністю ізолювати функціональні модулі системи TTMON від користувачів, які не пройшли первинну верифікацію. Завдяки реєстрації компонента у глобальних налаштуваннях фреймворку, перевірка відбувається автоматично на рівні ядра,

що унеможливлює обхід захисного екрана шляхом прямого введення URL-адреси аналітичного дашборду в рядок браузера. Візуальний вигляд інтерфейсу, який бачить новий співробітник агенції під час першої спроби авторизації в системі, представлено на рисунку 3.2.

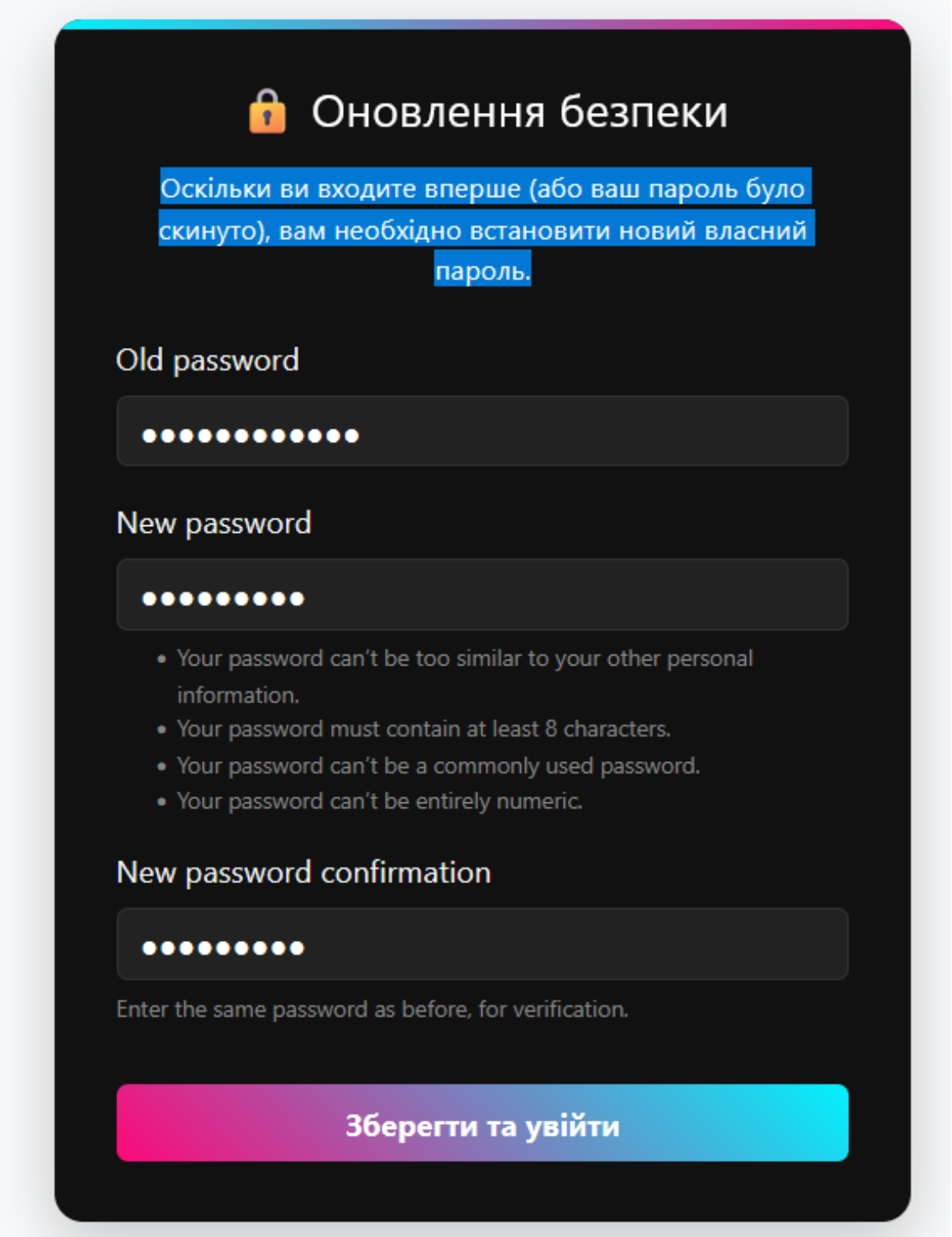

Рис. 3.2 - Інтерфейс примусової зміни пароля під час першого входу в систему

Після того як користувач вводить новий пароль, що відповідає встановленим критеріям складності, та успішно підтверджує форму, контролер змінює прапорець is_first_login на значення False. Це призводить до того, що під час обробки наступних запитів ForcePasswordChangeMiddleware пропускає сигнал далі по конвеєру безпосередньо до бізнес-логіки представлень. Створена підсистема автентифікації гарантує високий рівень безпеки корпоративних даних та забезпечує суворий контроль за дотриманням політики управління обліковими записами працівників маркетингової агенції.

3.3 Розробка модуля управління технічними завданнями

Серцевиною виробничого процесу в маркетинговій агенції є комунікація між фахівцем із закупівлі реклами та креативним дизайнером. Для цифровізації цього етапу було спроєктовано модуль управління технічними завданнями, який повністю замінює використання сторонніх месенджерів. Архітектура бази даних цього модуля базується на центральній сутності Task, яка зберігає всі метадані про майбутній відеоролик. Згідно з найкращими практиками проєктування моделей у Django, для полів із фіксованим набором значень було використано механізм кортежів choices, що оптимізує зберігання даних у базі та автоматично генерує коректні елементи інтерфейсу у формах [14, 113с.].

Для забезпечення ізоляції вхідних та вихідних матеріалів архітектура бази даних передбачає дві додаткові моделі: TaskFile для збереження референсів від баєра та TaskResultFile для завантаження готових креативів дизайнером. Завдяки використанню зовнішніх ключів система дозволяє прикріплювати необмежену кількість медіафайлів до одного технічного завдання. Частковий програмний код, який демонструє структуру базової моделі та набір статусів, наведено у лістингу 3.6.

Лістинг 3.6 Структура моделі даних технічного завдання

```
from django.db
import models from django.conf
import settings from analytics.models
import AdCreative
```

```
class Task(models.Model):
  STATUS_CHOICES = ( ('DRAFT', 'Чернетка'), ('SENT', 'Відправлено'),
  ('IN_PROGRESS', 'В роботі'), ('REVIEW', 'На перевірці'), ('COMPLETED',
  'Готово'), )
title = models.CharField(max_length=255, verbose_name="Назва ТЗ")
author = models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE, related_name='created_tasks')
assignee = models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.SET_NULL, null=True, blank=True)
source_creative = models.ForeignKey(AdCreative,
on_delete=models.SET_NULL, null=True, blank=True)
status = models.CharField(max_length=20, choices=STATUS_CHOICES,
default='DRAFT')
deadline = models.DateField(verbose_name="Дедлайн", null=True,
blank=True)
needs_voiceover = models.BooleanField(default=False)
created_at = models.DateTimeField(auto_now_add=True)
creative_comment = models.TextField(verbose_name="Коментар", blank=True,
null=True)
```

Керування життєвим циклом завдання реалізовано за принципом кінцевого автомата State Machine. Контролер обробки деталей завдання task_detail перевіряє права доступу поточного користувача до конкретного об'єкта та забезпечує маршрутизацію дій залежно від натиснутої кнопки в інтерфейсі. Логіка контролера жорстко регламентує, які статуси можуть бути змінені. Коли дизайнер завантажує готові файли та натискає кнопку відправки, контролер перехоплює дію submit_result та автоматично переводить завдання у статус перевірки. З цього моменту відповідальність переходить до медіабаєра, який може або прийняти роботу, або повернути її на доопрацювання з відповідним коментарем. Програмну реалізацію цієї бізнес-логіки представлено у лістингу 3.7.

Лістинг 3.7 Обробник зміни статусів життєвого циклу завдання

```
@login_required
def task_detail(request, pk):
    task = get_object_or_404(Task, pk=pk)
    if request.method == 'POST':
        action = request.POST.get('action')

        if action == 'submit_result':
            task.creative_comment = request.POST.get('creative_comment')
            task.status = 'REVIEW'
            task.save()
            files = request.FILES.getlist('result_files')
            for f in files:
                TaskResultFile.objects.create(task=task, file=f)

        elif action == 'review_result':
            decision = request.POST.get('decision')
            if decision == 'approve':
                task.status = 'COMPLETED'
                task.save()
            elif decision == 'reject':
                task.status = 'IN_PROGRESS'
                task.buyer_feedback = request.POST.get('buyer_feedback')
                task.save()
        return redirect('tasks:task_detail', pk=task.id)
    return render(request, 'tasks/task_detail.html', {'task': task})
```

Перевагою такої програмної реалізації є повний контроль над процесом виробництва на рівні сервера. Клієнтська частина платформи формує візуальні елементи управління виключно на основі поточного статусу завдання та ролі користувача, що виключає можливість обходу системи шляхом підміни

параметрів запиту у браузері. Візуалізацію загального списку завдань із відображенням їхніх поточних статусів у системі наведено на рисунку 3.3.

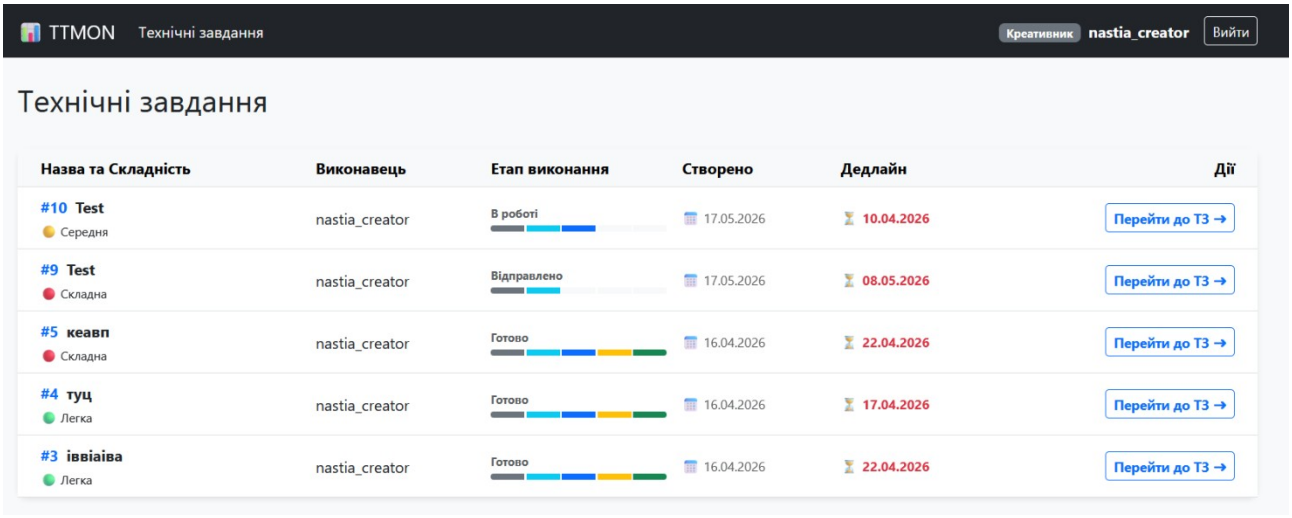

Рис. 3.3 - Інтерфейс модуля управління технічними завданнями платформи TTMON

Завдяки відокремленню етапів розробки контенту тімліди отримують можливість автоматично генерувати звіти про ефективність кожного дизайнера, відслідковуючи середній час перебування технічного завдання у статусі виконання.

3.4 Програмування процесу інтеграції з TikTok API

Автоматизація отримання статистичних даних з рекламних кабінетів є ключовим технічним завданням платформи TTMON. Для реалізації цього механізму в системі використовується офіційний протокол авторизації OAuth 2.0, який дозволяє отримувати безпечний доступ до даних облікових записів без збереження паролів на стороні нашого сервера [13, 252с.]. З точки зору програмної архітектури, процес інтеграції поділений на два етапи:

перенаправлення користувача на сторінку підтвердження дозволів у системі TikTok та подальший обмін тимчасового коду на постійні токени доступу Access та Refresh. Ця логіка обробляється у модулі users за допомогою двох контролерів: tiktok_login для ініціалізації процесу та tiktok_callback для приймання відповіді від серверів соціальної мережі.

Коли системний адміністратор ініціює підключення нового Business Center, система генерує унікальний криптографічний рядок стану, який зберігається у сесії користувача для захисту від атак підробки міжсайтових запитів CSRF. Після того як користувач надає необхідні дозволи на стороні TikTok, система перенаправляє його назад на вказаний маршрут callback разом із тимчасовим кодом. Контролер перевіряє цілісність параметра стану і, у разі успішної перевірки, формує POST-запит до програмного інтерфейсу платформи за допомогою вбудованої бібліотеки requests (лістинг 3.8).

Лістинг 3.8 Програмна реалізація контролера обробки відповіді від TikTok

```
@login_required
def tiktok_callback(request):
code = request.GET.get('code')
        state = request.GET.get('state') if state != request.session.get('tiktok_state'):
        messages.error(request, "Security error: invalid state.")
        return
redirect('users:buyer_integrations')
token_url =
"[https://open.tiktokapis.com/v2/oauth/token/](https://open.tiktokapis.com/v2/oauth/
token/)"
      data = {
         'client_key': settings.TIKTOK_CLIENT_KEY,
         'client_secret': settings.TIKTOK_CLIENT_SECRET,
         'code': code,
         'grant_type': 'authorization_code',
         'redirect_uri': settings.TIKTOK_REDIRECT_URI,
      }
```

```
response = requests.post(token_url, data=data).json()

if 'access_token' in response:
    TikTokIntegration.objects.update_or_create(
        user=request.user,
        defaults={
            'open_id': response['open_id'],
            'access_token': response['access_token'],
            'refresh_token': response.get('refresh_token', ''),
            'display_name': 'TikTok Account'
        }
    )
```

Успішна відповідь, яка містить токени доступу, негайно обробляється механізмом об'єктно-реляційного відображення, що створює або оновлює відповідний запис у таблиці TikTokIntegration. Модель бази даних для збереження інтеграцій передбачає не лише фіксацію криптографічних ключів, а й зв'язки з внутрішньою організаційною структурою агенції. Зокрема, поле allowed_buyers реалізовано як зв'язок Many-to-Many, що дозволяє адміністратору гнучко призначати права доступу до конкретного Business Center для різних медіабаєрів, не надаючи їм прямого контролю над обліковим записом.

Логіка управління правами доступу до підключених рекламних кабінетів реалізована на рівні представлення. Контролер buyer_integrations розділяє користувачів на дві категорії: адміністратори мають повний доступ до управління всіма інтеграціями та можуть призначати баєрів на конкретні кабінети, тоді як звичайні медіабаєри бачать лише ту статистику, яка закріплена безпосередньо за ними. Завдяки використанню комплексної фільтрації через об'єкт django.db.models.Q система ефективно витягує з бази даних лише релевантну інформацію. Візуальний інтерфейс модуля управління інтеграціями та розподілу прав доступу між співробітниками представлено на рисунку 3.4.

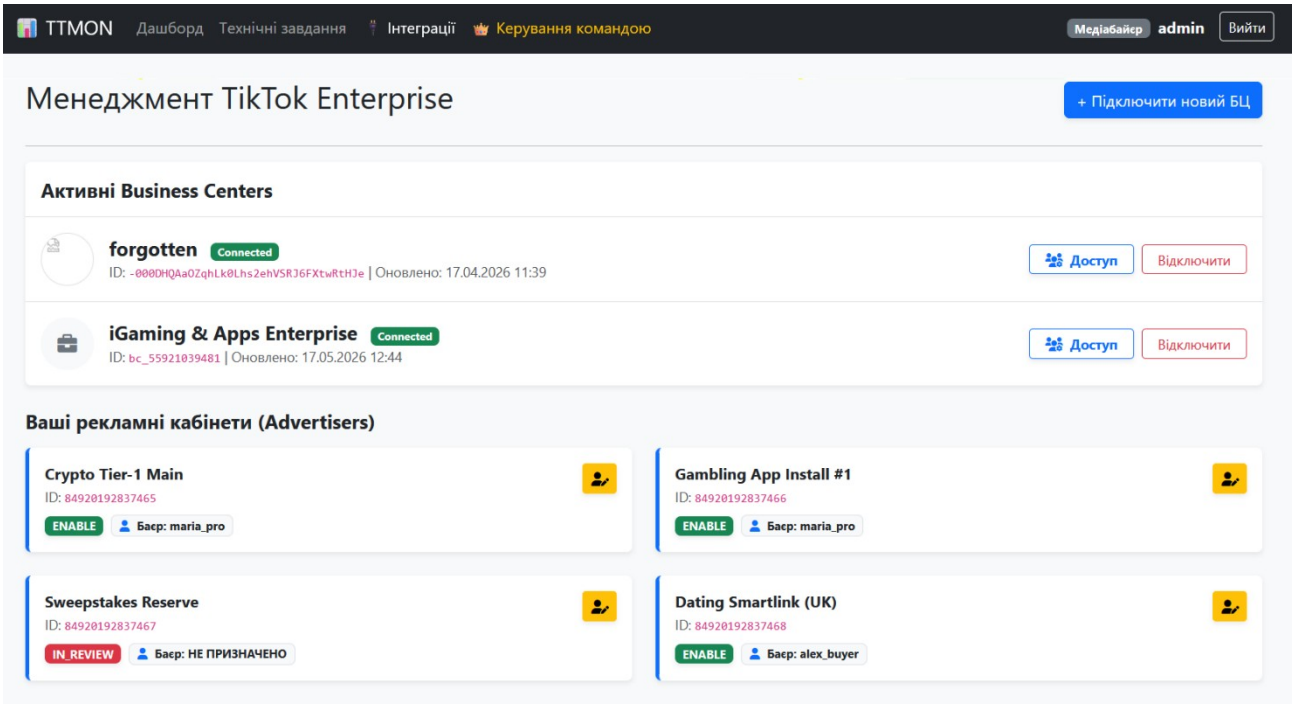

Рис. 3.4 - Інтерфейс підключення рекламних кабінетів та налаштування доступу

Після успішного отримання та безпечного збереження ключів доступу, система TTMON переходить у режим фонової агрегації статистичних даних, використовуючи ці токени для регулярного формування запитів до захищених маршрутів Marketing API, що забезпечує безперервність процесу моніторингу ефективності креативів

3.5 Створення аналітичного двигуна та алгоритму виявлення вигорання

Отримання сирих статистичних даних із програмних інтерфейсів рекламних платформ є лише першим етапом моніторингу. Для того щоб медіабаєри могли приймати оперативні рішення, ці масиви інформації повинні бути миттєво оброблені, агреговані та перетворені на ключові показники ефективності. З цією метою у додатку analytics було розроблено власний аналітичний двигун, побудований на базі інструментів об'єктно-реляційного відображення фреймворку Django. Використання вбудованих функцій агрегації бази даних дозволяє перенести складні математичні обчислення з оперативної пам'яті веб-

сервера безпосередньо на рівень СУБД PostgreSQL, що критично важливо для збереження швидкодії системи при роботі з тисячами рекламних креативів [14, 84с.].

Фундаментальні показники ефективності розраховуються динамічно на рівні моделі Campaign. Замість того, щоб зберігати статичні значення коефіцієнта клікабельності або середнього показника рентабельності інвестицій у базі даних, система обчислює їх під час кожного звернення до об'єкта за допомогою декоратора властивостей [14, 84с.]. Особливу увагу в моделі приділено алгоритму виявлення вигорання рекламної кампанії. Цей процес охоплює повний цикл обчислень: від аналізу індивідуальної ефективності кожного креативу на основі співвідношення конверсій до витрат із врахуванням мінімального порогу статистичної значущості, до фінальної агрегації відсотка вигорілих матеріалів у межах усієї кампанії. Математично ця комплексна логіка описується наступною системою рівнянь:

$$R_{bo} = \left(\frac{\sum_{i=1}^{N} B_i}{N} \right) \cdot 100\%, \tag{3.1}$$

де $B_i = \begin{cases} 1, & якщо \ \frac{Conv_i}{Spend_i} < \theta \ та \ Spend_i \geq S_{min} \\ 0, & інакше \end{cases}$

У цій математичній моделі R_{bo} позначає фінальний відсоток вигорання рекламної кампанії, який розраховується як середнє арифметичне значення вигорілих креативів відносно їх загальної кількості N. Ключовим елементом формули є змінна B_i, яка виступає бінарним індикатором стану для кожного окремого *і-го* відеоролика. Цей індикатор працює за принципом строгої умовної логіки і набуває значення одиниці, що алгоритмічно підтверджує факт вигорання, виключно за одночасного виконання двох критичних критеріїв. Перша умова вимагає, щоб поточна рентабельність конкретного креативу, яка розраховується як відношення фактично отриманих конверсій до витраченого бюджету, впала нижче заздалегідь встановленого порогу ефективності θ. Друга умова гарантує статистичну достовірність аналітики: сумарні витрати на цей конкретний креатив повинні обов'язково перевищити мінімально необхідний

фінансовий ліміт S_{min}. Такий двоетапний підхід повністю унеможливлює хибне маркування нових рекламних матеріалів, які щойно були запущені та ще не пройшли обов'язковий етап первинного машинного навчання алгоритмів соціальної мережі. Якщо хоча б один із цих параметрів не відповідає заданим критеріям, індикатор B_i залишається нульовим, і креатив продовжує вважатися активним. Програмна реалізація цих комплексних обчислень у середовищі Python представлена у лістингу 3.9.

Лістинг 3.9 Програмна реалізація динамічних розрахунків KPI на рівні моделі

```
class Campaign(models.Model):
budget = models.DecimalField(max_digits=10, decimal_places=2, verbose_name="Бюджет ($)")
spend = models.DecimalField(max_digits=10, decimal_places=2, default=0, verbose_name="Витрачено ($)")
impressions = models.IntegerField(default=0, verbose_name="Покази")
clicks = models.IntegerField(default=0, verbose_name="Кліки")
conversions = models.IntegerField(default=0, verbose_name="Конверсії")
@property
def ctr(self):
   return round((self.clicks / self.impressions * 100), 2) if self.impressions > 0 else 0
@property
def burnout_rate(self):
   creatives = self.creatives.all()
   if not creatives: return 0
   burning = creatives.filter(is_burning_out=True).count()
   return round((burning / creatives.count()) * 100, 1)
@property
def average_roi(self):
   avg = self.creatives.aggregate(Avg('roi'))['roi__avg']
   return round(avg, 2) if avg is not None else 0
```

Логіка математичного виявлення вигорання індивідуального відеоролика базується на безперервному порівнянні його поточних фінансових метрик з історичними максимумами та заданими еталонними порогами. Якщо алгоритм фіксує, що динаміка конверсій різко падає при збереженні або зростанні щоденних витрат, система автоматично перемикає булевий прапорець is_burning_out у стан True для конкретного об'єкта.

Для формування зведеного дашборду керівника або медіабаєра рівень представлення здійснює глобальну агрегацію даних за допомогою функцій Sum, Count та Avg. Контролер dashboard не лише розраховує загальні витрати за всіма активними кампаніями, але й автоматично визначає найефективніший креатив та найкращого фахівця агенції на основі кількості згенерованих конверсій. Застосування спеціального об'єкта запитів дозволяє будувати складні багаторівневі фільтри, які моментально реагують на вибір географії, статусу або конкретного учасника команди (лістинг 3.10).

Лістинг 3.10 Глобальна агрегація статистики для аналітичного дашборду

```
@login_required
def dashboard(request):
queryset = Campaign.objects.all().prefetch_related('creatives')
stats_queryset = queryset.filter(id__in=selected_campaign_ids) if selected_campaign_ids else queryset
stats = {
    'total_spend': stats_queryset.aggregate(Sum('spend'))['spend__sum'] or 0,
    'active_campaigns': stats_queryset.filter(status='ACTIVE').count(),
    'burning_campaigns': AdCreative.objects.filter(
        campaign__in=stats_queryset, is_burning_out=True
    ).values('campaign').distinct().count(),
    'top_creative': AdCreative.objects.filter(
        campaign__in=stats_queryset
    ).order_by('-roi').first(),
    'top_buyer': CustomUser.objects.filter(
        id__in=stats_queryset.values('owner')
```

```
                ).annotate(total_conv=Sum('campaigns__conversions')).order_by('-
total_conv').first(),
        'avg_roi': AdCreative.objects.filter(
            campaign__in=stats_queryset
        ).aggregate(Avg('roi'))['roi__avg'] or 0
    }
```

Отримана аналітична статистика передається до клієнтської частини додатка, де перетворюється на інтерактивні графіки та віджети. Це дозволяє співробітникам агенції здійснювати глибокий моніторинг рекламної активності в режимі реального часу, незалежно від кількості підключених бізнес-центрів. Візуалізацію роботи аналітичного двигуна у користувацькому інтерфейсі з відображенням агрегованих метрик та графіків наведено на рисунку 3.5.

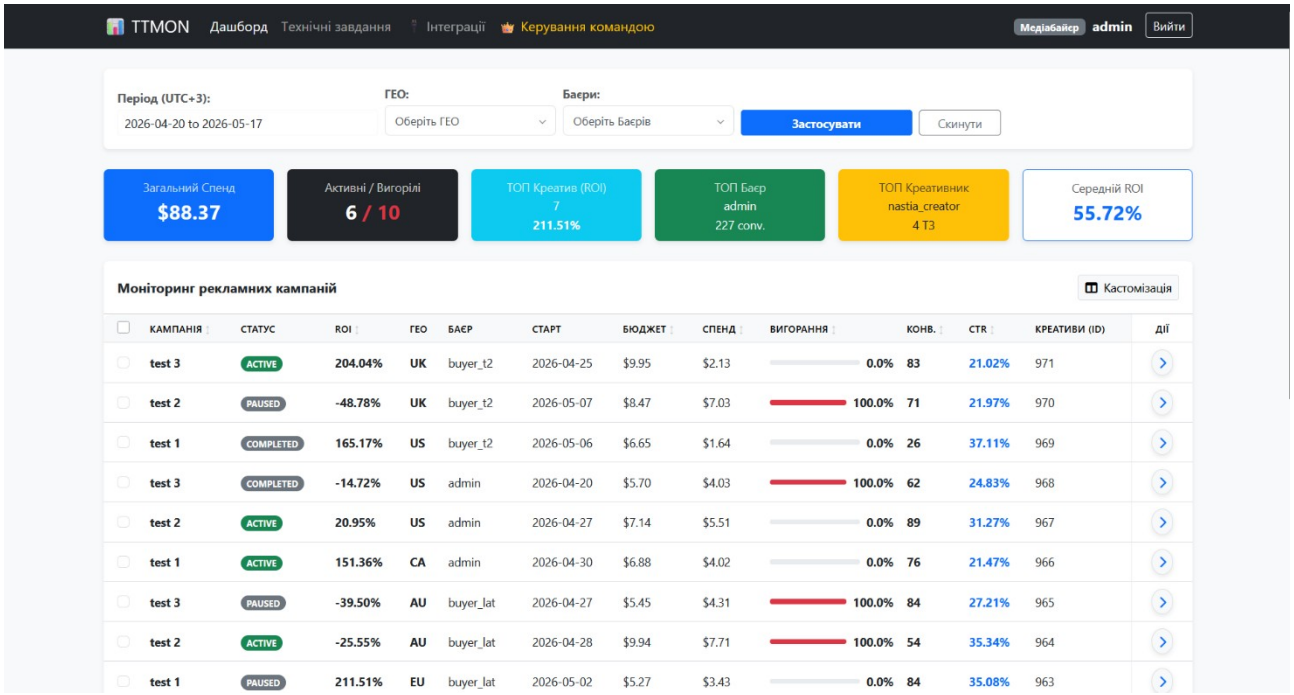

Рис. 3.5 - Аналітичний дашборд рекламного креативу з відображенням агрегованої статистики

Розроблений механізм обробки даних гарантує високу точність фінансових показників та дозволяє агенції своєчасно реагувати на неефективні рекламні зв'язки, суттєво заощаджуючи маркетинговий бюджет клієнтів.

3.6 Розробка клієнтської частини

Оскільки система TTMON орієнтована на інтенсивну взаємодію користувачів із великими масивами аналітичних даних та мультимедійним контентом, класичного підходу із повним перезавантаженням сторінок виявилося недостатньо для забезпечення сучасного рівня користувацького досвіду. Тому клієнтська частина платформи була суттєво розширена за рахунок використання технологій асинхронного програмування та маніпуляцій з об'єктною моделлю документа DOM за допомогою JavaScript. Логіку фронтенду було розділено на два ключові напрямки: інтерактивна візуалізація статистики та динамічне управління медіафайлами під час постановки технічних завдань.

3.6.1 Динамічні дашборди

Серцевиною робочого простору медіабаєра є аналітична таблиця, яка агрегує фінансові показники за всіма активними рекламними кампаніями. Оскільки різні фахівці фокусуються на різних метриках (наприклад, керівника цікавить загальний ROI та Спенд, а дизайнера – рівень вигорання та CTR), виникла архітектурна необхідність надати користувачам можливість самостійно кастомізувати вигляд аналітичного звіту.

Для вирішення цього завдання було інтегровано легкоту бібліотеку Sortable.js, яка дозволяє реалізувати механізм перетягування Drag-and-Drop безпосередньо в браузері. В інтерфейсі дашборду було створено спеціальне модальне вікно кастомізації, де користувач може змінювати порядок відображення колонок таблиці та приховувати непотрібні метрики. Для того щоб ці налаштування не втрачалися після оновлення сторінки, логіка клієнтської частини використовує локальне сховище браузера.

Функція saveTableSettings зчитує поточний стан перемикачів та послідовність елементів у списку, формує масив конфігураційних об'єктів у форматі JSON і зберігає його в пам'яті клієнтського пристрою. При кожному наступному завантаженні сторінки браузер автоматично викликає функцію

applyTableSettings, яка динамічно перебудовує структуру HTML-таблиці відповідно до збережених преференцій користувача, фізично переміщуючи вузли за допомогою методу appendChild. Програмну реалізацію цього механізму наведено у лістингу 3.11.

Лістинг 3.11 Логіка кастомізації та збереження стану таблиці

```
const orderList = document.getElementById('column-order-list'); Sortable.create(orderList, { animation: 150 });
function saveTableSettings() { const settings = []; orderList.querySelectorAll('li').forEach(li => { settings.push({ id: li.getAttribute('data-id'), visible: li.querySelector('.col-toggle').checked }); }); localStorage.setItem('ttmon_table_v5', JSON.stringify(settings)); location.reload(); }
function applyTableSettings() { const cfg = JSON.parse(localStorage.getItem('ttmon_table_v5')); if (!cfg) return; const headerRow = document.getElementById('table-header-row'); const rows = document.querySelectorAll('#campaignTable tbody tr');
cfg.forEach(item => {
  const th = document.querySelector(`th[data-id="${item.id}"]`);
  if (th) {
    headerRow.appendChild(th);
    th.style.display = item.visible ? '' : 'none';
  }
  rows.forEach(row => {
    const td = row.querySelector(`td[data-id="${item.id}"]`);
    if (td) {
      row.appendChild(td);
      td.style.display = item.visible ? '' : 'none';
    }
  });
} window.addEventListener('DOMContentLoaded', applyTableSettings);
```

Окрім кастомізації структури, таблицю було оснащено механізмом двостороннього клієнтського сортування. Замість того, щоб щоразу відправляти

запит на сервер для сортування даних за зростанням чи спаданням бюджету, JavaScript-алгоритм миттєво переставляє рядки таблиці в оперативній пам'яті браузера. Алгоритм автоматично очищає значення від символів валют та відсотків, перетворює їх на числа з плаваючою комою та здійснює сортування типу "бульбашка". Це значно знижує навантаження на серверну інфраструктуру та забезпечує миттєвий відгук інтерфейсу. Візуальний вигляд готового аналітичного дашборду представлено на рисунку 3.6.

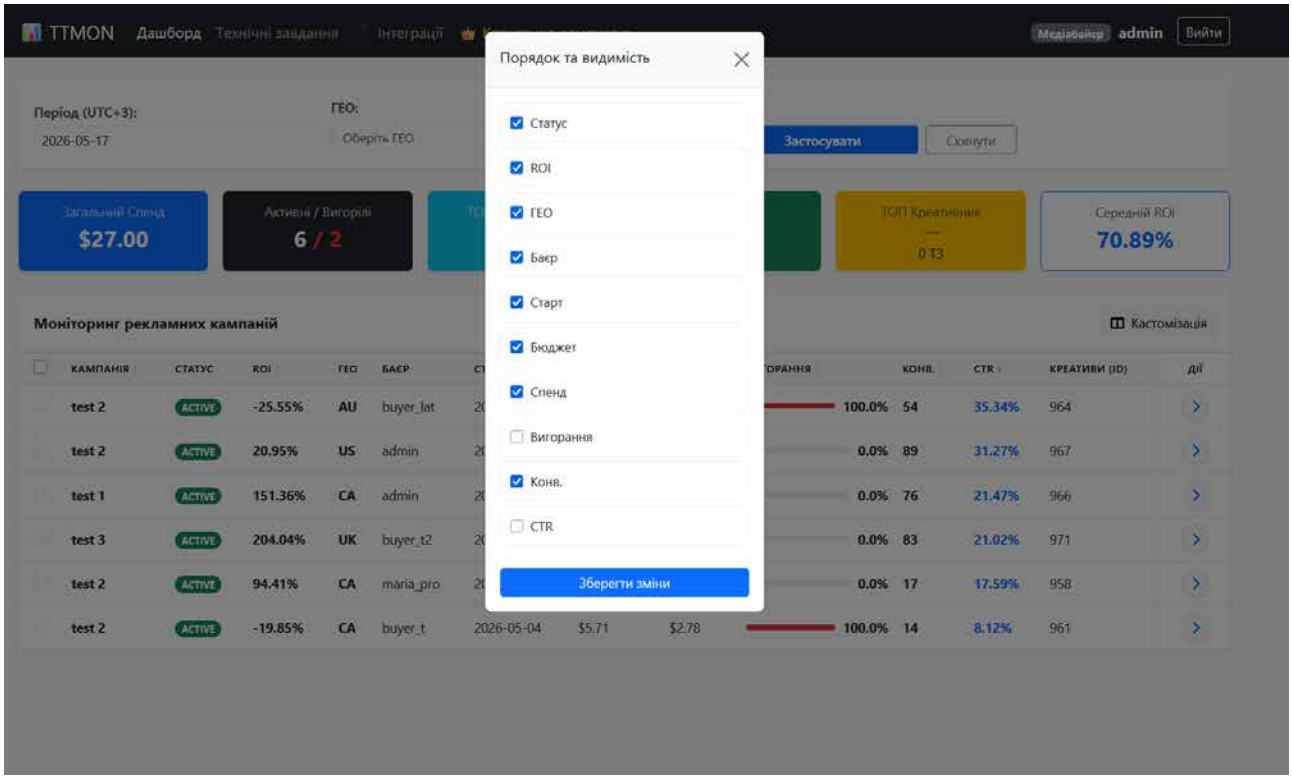

Рис. 3.6 - Інтерфейс головного аналітичного дашборду з динамічними графіками та кастомізованою таблицею

3.6.2 Live Preview

Процес створення технічного завдання для дизайнерів вимагає завантаження великої кількості референсних матеріалів. Стандартна поведінка HTML-елемента типу input type="file" має суттєвий недолік: об'єкт FileList, який зберігає вибрані файли, є доступним лише для читання. Це означає, що користувач не може видалити один конкретний файл зі списку вибраних перед

відправкою форми на сервер – йому доведеться вибирати всі файли наново. Крім того, стандартна форма не дозволяє переглянути відео до моменту його фізичного завантаження у базу даних.

Для вирішення цієї інженерної проблеми клієнтську архітектуру платформи TTMON було модернізовано з використанням сучасного програмного інтерфейсу DataTransfer API [15]. Цей інтерфейс дозволяє створювати віртуальні контейнери для файлів у пам'яті браузера, вільно додавати або видаляти об'єкти з них, а потім програмно підміняти вміст прихованого HTML-інпуту безпосередньо перед відправкою форми на бекенд.

Паралельно з накопиченням файлів у об'єкті DataTransfer, система реалізує технологію Live Preview - попередній перегляд у реальному часі. Для кожного доданого медіафайлу викликається вбудована функція браузера URL.createObjectURL(). Вона генерує тимчасове локальне посилання, яке вказує безпосередньо на файл у файловій системі комп'ютера користувача або в оперативній пам'яті, минаючи інтернет-з'єднання. Це посилання миттєво підставляється в атрибут src динамічно створеного тегу <video> або <img>, що дозволяє медіабаєру відтворити ролик прямо у формі створення ТЗ. Логіка роботи цього механізму відображена у лістингу 3.12.

Лістинг 3.12 Програмна реалізація DataTransfer API та Live Preview

```
let dataTransfer = new DataTransfer(); const MAX_FILES = 10;
$('#dummyFileInput').on('change', function(e) { for (let file of this.files) { if (dataTransfer.items.length >= MAX_FILES) break; dataTransfer.items.add(file); } this.value = ""; updateFilesUI(); });

function removeNewFile(index) { let newDataTransfer = new DataTransfer(); for(let i = 0; i < dataTransfer.items.length; i++){ if(i !== index) newDataTransfer.items.add(dataTransfer.items[i].getAsFile()); } dataTransfer = newDataTransfer; updateFilesUI(); }

function updateFilesUI() { $('#realFileInput')[0].files = dataTransfer.files; $('.new-file-item').remove(); $('.new-media-preview').remove();

for(let i = 0; i < dataTransfer.files.length; i++) {

    let file = dataTransfer.files[i];
```

```
let objUrl = URL.createObjectURL(file);
let mediaHtml = '';

if(file.type.startsWith('video/')) {
    mediaHtml = `<video src="${objUrl}" controls class="rounded w-100" style="max-height: 250px; background: #000;"></video>`;
} else if(file.type.startsWith('image/')) {
    mediaHtml = `<img src="${objUrl}" class="rounded w-100" style="max-height: 250px; object-fit: contain;">`;
}
$('#p-media-list').append(`
  <div class="mb-3 border p-2 rounded bg-success bg-opacity-10 new-media-preview">
                <div class="fw-bold small text-success mb-1">Файл: ${file.name}</div>
      ${mediaHtml}
  </div>
`);
```

Додатково до попереднього перегляду файлів, клієнтська частина забезпечує миттєву візуалізацію текстової інформації. Усі текстові поля та випадаючі списки Select2 у лівій частині форми "прослуховуються" за допомогою обробників подій on('input') та change. Як тільки користувач вводить символ або обирає нове ГЕО чи статус складності, JavaScript миттєво копіює ці дані у блок праворуч. Зовнішній вигляд форми введення даних наведено на рисунку 3.7.

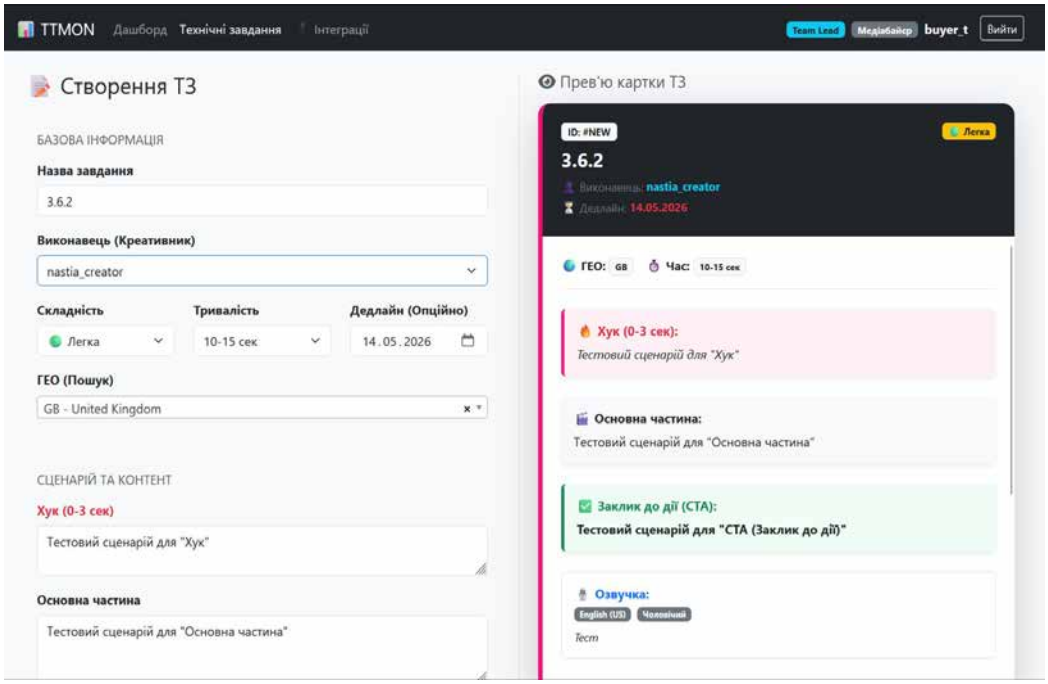

Рис. 3.7 - Інтерфейс форми створення технічного завдання з інтерактивними полями

Таким чином, медіабаєр ще до моменту збереження ТЗ бачить фінальну картку завдання рівно в такому ж вигляді, в якому її отримає відповідальний креативний дизайнер. Це дозволяє уникнути помилок форматування, перевірити коректність обрізки тексту та переконатися в якості завантажених референсних відеороликів. Інтерфейс модуля Live Preview з інтегрованим медіаплеєром продемонстровано на рисунку 3.8.

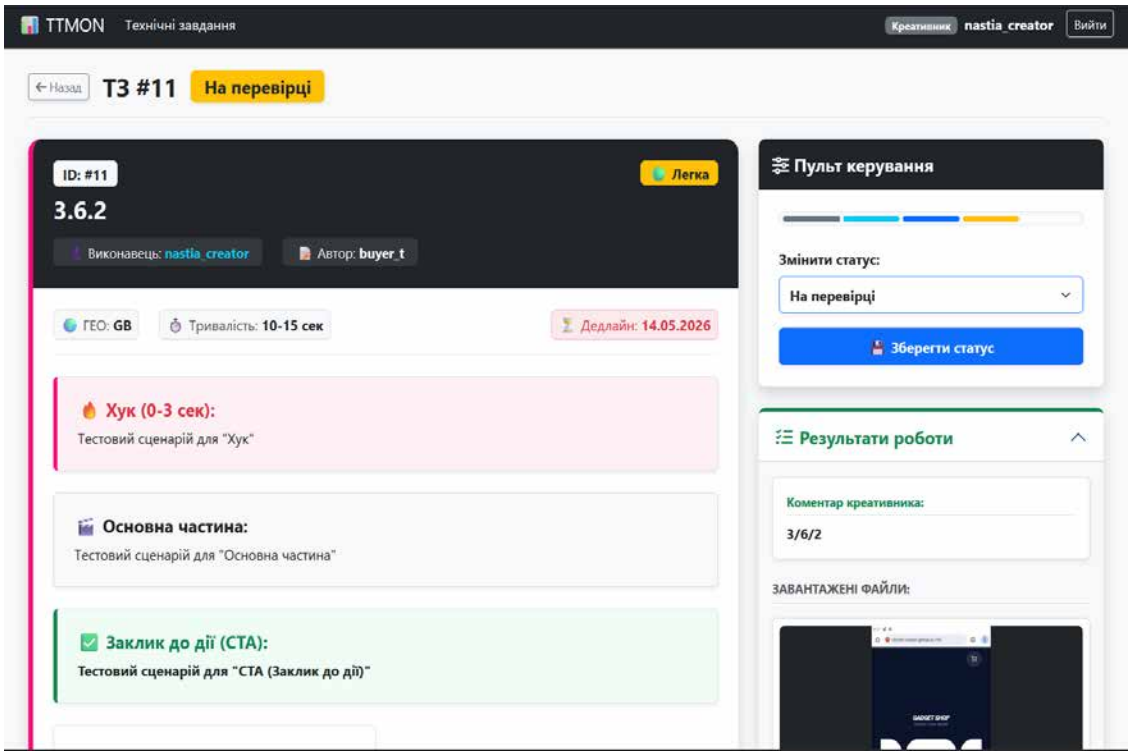

Рис. 3.8 - Вікно попереднього перегляду Live Preview із відтворенням завантаженого відеоролика

Реалізація клієнтської частини з використанням асинхронних оновлень DOM, локального сховища та DataTransfer API перетворила статичні HTML-сторінки на повноцінний інтерактивний веб-додаток, що значно прискорило виробничі процеси всередині маркетингової агенції та зменшило навантаження на сервер.

4 ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ СИСТЕМИ

4.1 Інфраструктурне забезпечення та конфігурація середовища розгортання

Заключним етапом інженерного життєвого циклу проектування та програмної реалізації інформаційної системи моніторингу TTMON є її комплексне функціональне тестування, аналіз стійкості архітектурних рішень та підготовка базової інфраструктури до подальшого виробничого впровадження. Процес розгортання сучасних корпоративних веб-застосунків, інтегрованих із зовнішніми програмними інтерфейсами, вимагає створення ізольованого середовища розробки. Це необхідно для забезпечення стабільності кодової бази, контролю версій, мінімізації ризиків втрати конфіденційних даних та забезпечення безперебійної двосторонньої взаємодії між сервером застосунків та хмарними серверами авторизації рекламних платформ. Інфраструктурний комплекс системи було розбито на три взаємопов'язані сегменти: децентралізоване керування версіями, конфігурація захищених мережевих тунелів для обробки інтеграційних викликів та делегування публічних доменних зон.

Для організації прозорого процесу написання коду, фіксації проміжних етапів розробки та забезпечення надійного резервного копіювання було застосовано децентралізовану систему контролю версій Git. Як єдине віддалене сховище для спільної роботи та ведення репозиторію було обрано хмарну платформу GitHub. У межах платформи було розгорнуто приватний репозиторій TTMON, архітектура якого будувалася за класичною моделлю Git Flow. Візуальну структуру організації кодової бази та журналу фіксації змін на віддаленому сервері GitHub наведено на рисунку 4.1.

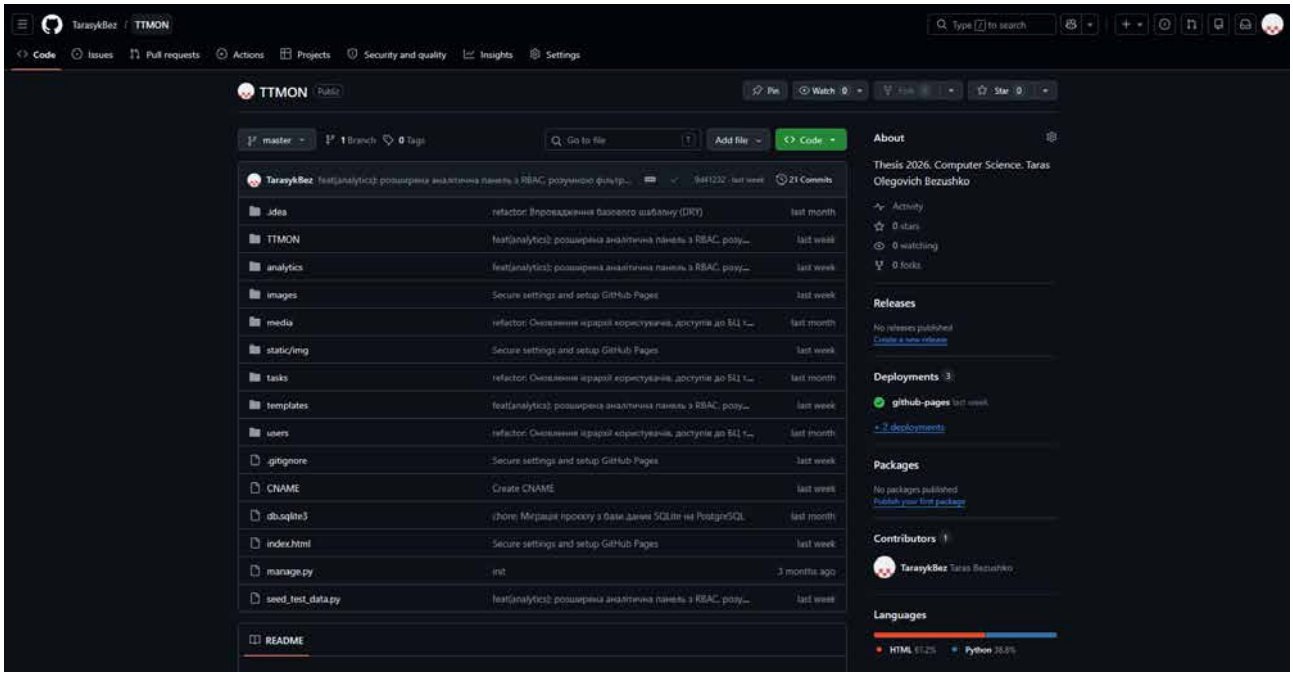

Рис. 4.1 – Структура віддаленого репозиторію проєкту TTMON на платформі GitHub

Найбільш складним інженерним завданням на етапі інфраструктурного моделювання стала організація стабільного зв'язку з серверами TikTok Marketing API для відлагодження протоколу OAuth 2.0. Специфіка сучасних закритих комерційних API полягає у суворих вимогах до безпеки: передача тимчасових криптографічних кодів авторизації та маркерів доступу на сервери розробника дозволена виключно через захищений протокол шифрування HTTPS. Під час ведення розробки на локальній робочій машині вбудований веб-сервер фреймворку Django за замовчуванням функціонує через незахищений протокол HTTP. Це повністю унеможливлює тестування механізмів зворотного виклику та веб-хуків від TikTok, оскільки зовнішні сервери безпеки блокують будь-які спроби передачі маркерів у незашифровані мережеві зони.

Для подолання цього обмеження та побудови відмовостійкого середовища тестування, повністю ідентичного промисловим серверам, було використано спеціалізовану утиліту мережевого тунелювання ngrok. Цей інструмент виступає у ролі зворотного проксі-сервера. Програма запускається локально та встановлює захищене з'єднання з глобальною хмарною архітектурою ngrok, автоматично виділяючи розробнику публічну URL-адресу з діючим цифровим

сертифікатом шифрування SSL/TLS. Будь-який зовнішній пакет даних, що надходить із мережі Інтернет на цю адресу, моментально проксирується хмарою всередину локальної мережі на вказаний порт сервера Django. Активний стан сформованого криптографічного тунелю та параметри моніторингу трафіку в реальному часі відображено на рисунку 4.2.

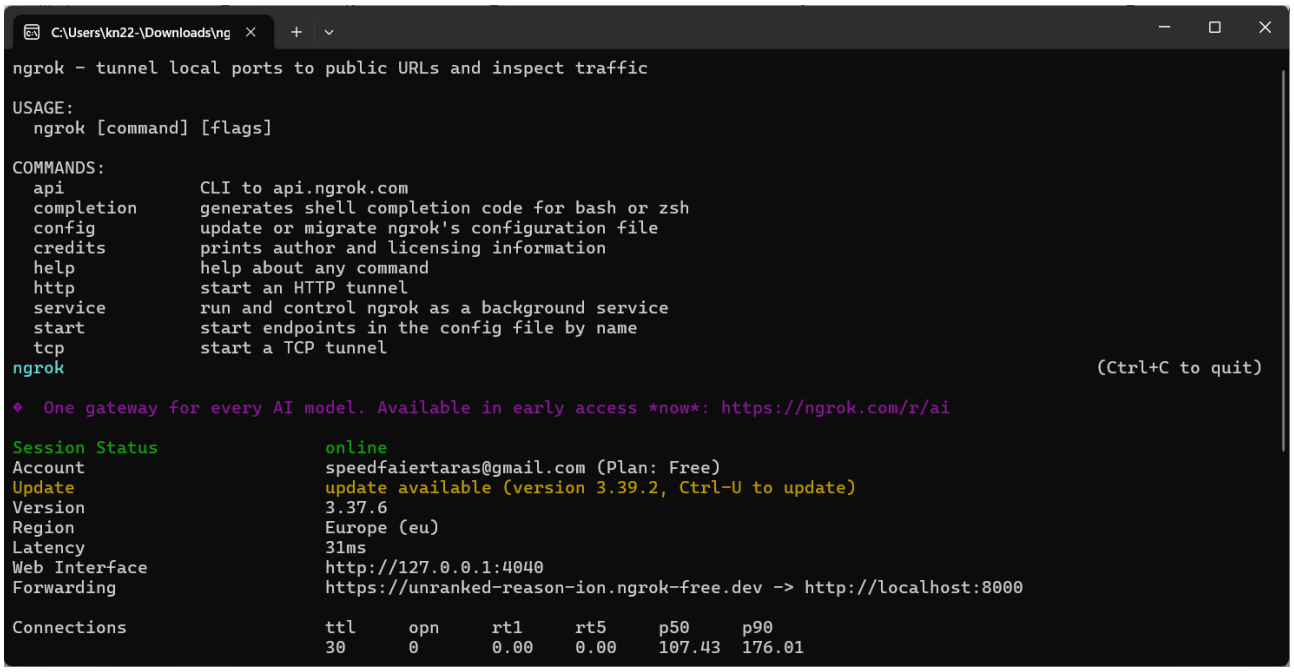

Рис. 4.2 – Конфігурація мережевого шлюзу утиліти ngrok для локального сервера

Організація такого наскрізного шлюзу вимагає внесення відповідних корективів до конфігураційних файлів ядра фреймворку з метою захисту від вразливостей, пов'язаних із підміною хостів. Згідно з концепціями ізоляції конфігураційного шару та управління змінними середовища [16, 62с.], отриману адресу тунелю було внесено до прихованого локального файлу конфігурації. У файлі settings.py було модифіковано перелік дозволених до обслуговування хостів, додано адресу до білого списку довірених джерел походження запитів для успішного проходження CSRF-валідації та перевизначено кінцеву точку обробки токенів TikTok (лістинг 4.1).

Лістинг 4.1 Конфігурація хостів та інтеграційного маршруту у файлі settings.py

```
ALLOWED_HOSTS = ['*', 'unranked-reason-ion.ngrok-free.dev']
CSRF_TRUSTED_ORIGINS = ['https://unranked-reason-ion.ngrok-free.dev']
```

TIKTOK_REDIRECT_URI = 'https://unranked-reason-ion.ngrok-free.dev/users/tiktok/callback/'

Паралельно з налаштуванням захищеного каналу агрегації аналітичних даних було виконано розгортання та конфігурацію презентаційного шару інформаційної системи. Стартова веб-сторінка платформи, що містить опис функціональних переваг, технологічного стеку та інтерфейсу TTMON для потенційних клієнтів маркетингової агенції, була інтегрована з офіційним доменним ім'ям ttmon.site.

З метою оптимізації швидкості завантаження та розподілу мережевих ресурсів було реалізовано концепцію роздільного хостингу: легка статична сторінка-візитка була відокремлена від ресурсомісткого бекенд-додатка і розгорнута за допомогою інструменту GitHub Pages. На стороні панелі управління глобального реєстратора доменних імен було виконано детальне налаштування ресурсних DNS-записів. Для коректного делегування прав було прописано чотири серійні A-записи, які вказують на захищені IP-адреси серверів доставки контенту GitHub, та один CNAME-запис для канонічного імені субдомену "www".

Така конфігурація інфраструктури дозволила досягти максимальної швидкості рендерингу інтерфейсу першого екрана за рахунок географічного розподілу серверів кешування. При спробі авторизації або переходу до аналітичних інструментів система здійснює автоматичне безшовне перенаправлення користувача з домену ttmon.site на захищений тунель сервера застосунків, де виконується основна логіка розподілу ролей та обробки фінансових метрик. Створення такого стійкого та відмовостійкого інфраструктурного комплексу повністю завершило етап конфігурації середовища та дозволило перейти до безпосереднього проектування та виконання сценаріїв тестування.

4.2 Функціональне тестування модулів та перевірка безпеки

Важливим компонентом забезпечення якості та відмовостійкості розробленого програмного комплексу TTMON є проведення детального функціонального тестування та верифікація інтегрованих механізмів інформаційної безпеки. На етапі перевірки працездатності окремих модулів системи та їхньої взаємодії базовою методологією було обрано тестування поведінки застосунку на основі вхідних та вихідних даних без втручання у внутрішню архітектуру програмного коду. Цей підхід зосереджує увагу виключно на аналізі відповідності кінцевих результатів заданим параметрам та бізнес-вимогам агенції.

Вибір такої методології обґрунтований необхідністю симуляції реальної поведінки співробітників із різними рівнями технічної підготовки та посадовими обов'язками. У межах цього процесу було застосовано методи еквівалентного розбиття та аналізу граничних значень для формування репрезентативного набору тестових сценаріїв. Це дозволило перевірити не лише основні успішні сценарії, за яких користувач вводить коректні дані, а й реакцію веб-застосунку на некоректні, незавершені або потенційно шкідливі запити, що гарантує стабільність роботи серверної частини під час нестандартних навантажень.

Особливе місце у структурі функціонального контролю посідає верифікація механізмів розподілу прав доступу між користувачами різних категорій. Оскільки веб-платформа TTMON оперує конфіденційними комерційними даними маркетингової агенції, включаючи бюджети рекламних кампаній та фінансову рентабельність інвестицій, архітектурна ізоляція маршрутів є критично важливою для запобігання несанкціонованому витоку інформації. Тестування безпеки було спрямоване на перевірку поведінки системи при спробах перехресного доступу, коли користувач із обмеженими повноваженнями намагається вручну через адресний рядок браузера відкрити захищені контролери управління фінансами або адміністративні інтерфейси налаштування команди.

Перевірка ізоляції маршрутів виконувалася шляхом аналізу роботи інтегрованих декораторів фреймворку, а також кастомних фільтрів на рівні представлень. Тестування підтвердило, що при спробі неавторизованого

переходу на захищені аналітичні сторінки система безпеки автоматично блокує запит на етапі обробки проміжного програмного забезпечення та здійснює примусове перенаправлення користувача на сторінку автентифікації. Для авторизованих співробітників відділу дизайну було підтверджено коректність відсікання доступу до інтеграцій з рекламними кабінетами та автоматичного перенаправлення на персональний робочий простір зі списками завдань. Результати базового сценарію перевірки автентифікації наведено в таблиці 4.1.

Таблиця 4.1

Тестовий сценарій перевірки автентифікації та ізоляції маршрутів

Test Case ID	1	
Опис	Перевірити механізм ізоляції маршрутів, обробку некоректних даних та обов'язкову зміну тимчасового пароля під час першого входу в систему.	
Пріоритет	Критичний	
Передумови	В системі адміністратором створено нового користувача з роллю дизайнера та активним прапорцем першого входу. Відкрита сторінка авторизації.	
Кроки	**Дані тестування**	**Очікуваний результат**
1. Ввести валідні логін та тимчасовий пароль.	new_designer, temp123	Успішна первинна авторизація.
2. Спробувати перейти за прямим посиланням на сторінку дашборду.	/analytics/dashboard/	Система блокує перехід і примусово перенаправляє користувача на форму обов'язкової зміни пароля.
3. На формі зміни пароля ввести комбінацію, що складається лише з 4 цифр.	1234	Виведення системного повідомлення про помилку: пароль занадто короткий і не відповідає критеріям безпеки.
4. Ввести новий надійний пароль.	SecurePass!2026	Пароль успішно змінено, користувач автоматично перенаправляється на дозволену сторінку списку завдань

Наступним архітектурним блоком, що підлягав всебічній перевірці, став модуль управління життєвим циклом технічних завдань. Специфіка цього компонента полягає в реалізації складного кінцевого автомата, який контролює послідовну зміну статусів завантажених медіаматеріалів від початкової чернетки баєра до фінального затвердження та архівації. Функціональні тести цього модуля дозволили перевірити коректність обробки подій зміни станів об'єктів у базі даних, валідацію обов'язкових текстових полів (наприклад, наявності тексту

озвучки при активації відповідного перемикача) та стабільність механізму роботи з прикріпленими файлами на стороні клієнта за допомогою програмного інтерфейсу браузера.

Таблиця 4.2

Тестовий сценарій модуля управління технічними завданнями

Test Case ID	2	
Опис	Перевірити коректність зміни статусів технічного завдання від моменту створення чернетки до відхилення матеріалу на етапі перевірки.	
Пріоритет	Середній	
Передумови	Авторизовано користувача з правами створення завдань. Відкрита форма створення нового ТЗ. В базі даних існує щонайменше один виконавець.	
Кроки	**Дані тестування**	**Очікуваний результат**
1. Заповнити обов'язкові текстові поля без вказання відповідального виконавця та зберегти форму.	Назва: "Тестовий креатив 1", Опис: "Перевірка статусів".	Завдання успішно створюється в базі даних зі статусом "Чернетка".
2. Відкрити створене завдання, обрати виконавця зі списку та оновити дані.	Виконавець: creative_user.	Статус завдання автоматично змінюється на "Відправлено".
3. Авторизуватися як обраний виконавець, завантажити відео файл та натиснути кнопку відправки на перевірку.	Файл: promo_video.mp4	Медіафайл успішно завантажується на сервер, статус змінюється на "На перевірці".
4. Авторизуватися як керівник і відхилити завдання з коментарем.	Коментар: "Необхідно переробити хук на перших секундах".	Статус повертається на етап виконання, а введений коментар зберігається і стає видимим для дизайнера.

Найбільш відповідальним інженерним етапом тестування платформи стала перевірка математичного апарату аналітичного двигуна. На момент розробки та первинного налагодження системи інтеграція з TikTok Marketing API функціонувала у тестовому середовищі, що унеможливлювало отримання потоку реальних комерційних метрик. Для забезпечення повноцінного тестування розрахункових формул до переведення системи в промислову експлуатацію, аналітичний модуль було перевірено шляхом наповнення бази даних "сирими" даними.

За допомогою скриптів автоматизації та панелі адміністрування було згенеровано сотні тестових записів: рекламні кампанії з різними бюджетами,

зафіксованими витратами, показами та конверсіями. Використання цих сирих даних дозволило математично верифікувати коректність роботи динамічних властивостей моделі. Було підтверджено, що формули обчислення середньої рентабельності та складна дворівнева формула визначення рівня вигорання креативів працюють з абсолютною точністю. Алгоритм безпомилково перемикав прапорці деградації ефективності відеороликів за умови перетину встановлених критичних порогів.

Таблиця 4.3

Тестовий сценарій аналітичного двигуна на сирих даних

Test Case ID	3	
Опис	Перевірити коректність роботи алгоритму виявлення вигорання креативу та агрегації статистики на рівні рекламної кампанії.	
Пріоритет	Високий	
Передумови	Відключено зовнішні запити до TikTok API. У базу даних завантажено сирі тестові записи: кампанію з 5 підключеними рекламними креативами.	
Кроки	**Дані тестування**	**Очікуваний результат**
1. Встановити для одного з креативів суму витрат, що перевищує мінімальний ліміт аналізу.	Витрати: $60 (мінімум = $50)	Дані у базі успішно оновлено.
2. Встановити кількість конверсій для цього ж креативу, яка формує рентабельність, нижчу за критичний поріг.	Конверсії: 0 (рентабельність = 0%)	Дані у базі успішно оновлено.
3. Відкрити сторінку аналітичного дашборду.	URL-адреса: /analytics/dashboard/	Відбуваються обчислення і встановлення статуса «Вигоріло»
4. Перевірити глобальний показник рівня вигорання кампанії.	Загальна кількість креативів у кампанії: 5	Рівень вигорання: 20%

Проведене функціональне тестування підтвердило повну відповідність розробленої системи початковому технічному завданню. Логіка зміни статусів, захист маршрутів та математичний апарат аналітичного двигуна працюють стабільно та безвідмовно навіть за умови імітації стресових навантажень.

4.3 Аналіз продуктивності системи на основі тестових сценаріїв

Після підтвердження функціональної коректності та безпеки інформаційної системи TTMON наступним критично важливим етапом стало проведення аналізу її швидкодії. Оскільки розроблена платформа орієнтована на щоденне використання всередині маркетингової агенції, її центральний інтерфейсний модуль – аналітичний дашборд – повинен оперативно обробляти великі масиви інформації. У системах такого класу затримки під час генерації звітів безпосередньо знижують продуктивність роботи медіабаєрів та уповільнюють ухвалення рішень щодо оптимізації рекламних бюджетів. Тому головною метою цього етапу тестування було оцінювання швидкості рендерингу динамічних сторінок та мінімізація часу взаємодії додатка з базою даних під час виконання складних математичних розрахунків.

Під час проектування веб-застосунків на основі фреймворків, які використовують інструменти об'єктно-реляційного відображення, розробники часто стикаються з проблемою прихованої деградації швидкодії. Найбільш поширеним архітектурним недоліком є проблема N+1 запитів. Вона виникає тоді, коли обчислювальний алгоритм спочатку робить один первинний запит до бази даних для отримання списку основних об'єктів, наприклад, масиву рекламних кампаній. Потім, під час побудови інтерфейсної таблиці, система змушена виконувати окреме додаткове звернення для кожного рядка, щоб витягнути пов'язані сутності, такі як ім'я відповідального керівника або ідентифікатори підключених креативів. У результаті для відображення невеликої таблиці генеруються сотні дрібних запитів до системи управління базами даних, що призводить до критичного перевантаження каналів зв'язку та накопичення черг на стороні сервера.

Для усунення цієї проблеми та радикального прискорення обробки аналітичних даних на етапі розробки представлень було впроваджено механізм активного завантаження пов'язаних сутностей. Оптимізація бази даних була досягнута за допомогою комбінованого використання вбудованих методів інструменту об'єктно-реляційного відображення, а саме методів select_related та prefetch_related.

Метод select_related був інтегрований для оптимізації зв'язків типу «один-до-багатьох» на рівні бази даних. Під час формування запиту цей метод змушує фреймворк створювати єдиний складний запит із використанням оператора об'єднання таблиць бази даних. Це дозволило отримувати загальні дані рекламної кампанії та профіль її власника одночасно, повністю виключивши повторні звернення до жорсткого диска сервера.

У свою чергу, метод prefetch_related було застосовано для оптимізації зворотних зв'язків та зв'язків типу «багато-до-багатьох». Оскільки динамічний розрахунок рівня вигорання рекламних матеріалів та обчислення середнього коефіцієнта рентабельності інвестицій вимагають послідовного перебору всіх підключених до кампанії відеороликів, система виконує попереднє витягування даних. Фреймворк робить один додатковий запит для збору всіх релевантних креативів і здійснює їхнє логічне об'єднання безпосередньо в оперативній пам'яті сервера застосунків. Конфігурацію оптимізованого запиту в коді контролера дашборду наведено у лістингу 4.2.

Лістинг 4.2 Оптимізація звернень до бази даних у контролері представлення дашборду

```
queryset = Campaign.objects.all().select_related('owner').prefetch_related('creatives')
```

Впровадження цієї архітектурної оптимізації дозволило скоротити кількість операцій звернення до бази даних з потенційних кількох сотень до двох або трьох константних запитів, незалежно від обсягу інформації, що виводиться на екран користувача.

Для практичного оцінювання ефективності розроблених рішень було проведено тестування швидкості завантаження сторінок за допомогою панелі інструментів розробника у браузері. Тестовий сценарій передбачав повне оновлення аналітичного дашборду в умовах штучно створеного операційного навантаження, коли база даних містила понад сто активних кампаній та п'ятсот медіафайлів. Головними метриками під час замірів виступали час відповіді сервера до отримання першого байта контенту та сумарний час повної побудови об'єктної моделі документа в браузері клієнта.

Емпіричні вимірювання продемонстрували високий рівень швидкодії оптимізованої системи. Серверна обробка великої таблиці, що включає багаторівневу фільтрацію за географічними ознаками та ідентифікаторами користувачів, а також динамічне виконання формульних обчислень, триває в середньому від ста п'ятдесяти до двохсот п'ятдесяти мілісекунд. Клієнтська частина інтерфейсу миттєво інтерпретує отриманий масив даних, забезпечуючи плавне відображення індикаторів вигорання відео та віджетів загальної фінансової статистики. Результати успішного вимірювання часових показників завантаження системи зафіксовано на рисунку 4.3.

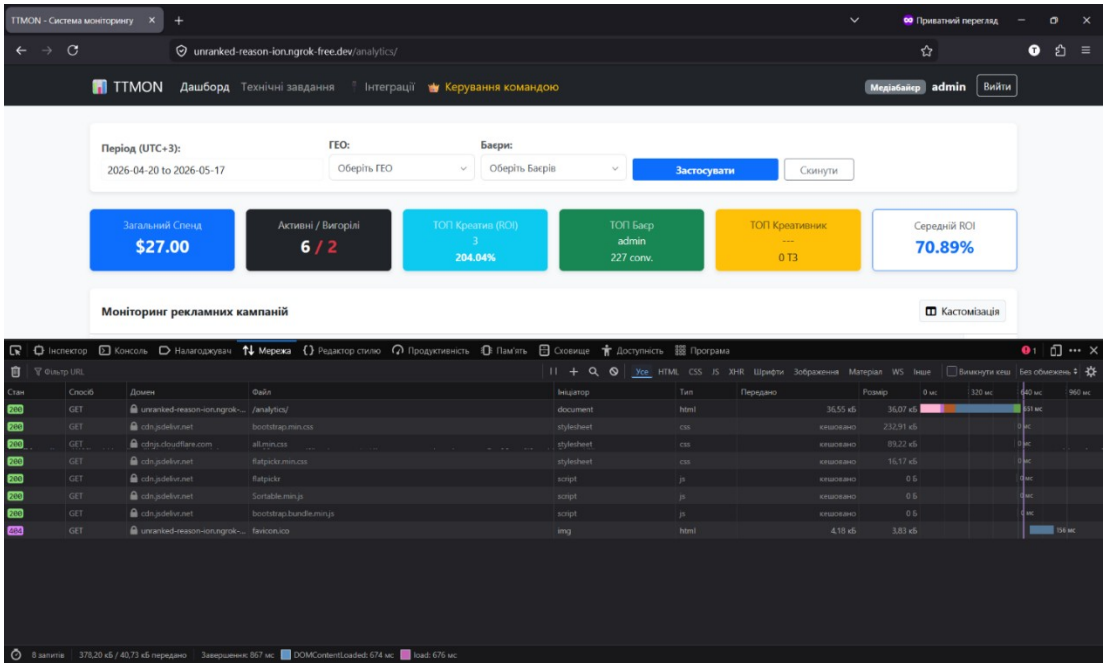

Рис. 4.3 – Результати вимірювання швидкості завантаження аналітичного дашборду в панелі розробника

Отримані під час тестування часові параметри доводять, що розроблена архітектура інформаційної системи моніторингу повністю готова до промислової експлуатації та подальшого масштабування. Застосування патернів активного завантаження даних та перенесення обчислень на рівень бази даних дозволило мінімізувати навантаження на апаратні ресурси сервера, гарантуючи стабільний та швидкий відгук інтерфейсу під час роботи з великими обсягами маркетингової аналітики.

4.4 Аналіз результатів впровадження та перспективи розвитку платформи

Завершення етапів інфраструктурного налаштування, функціонального тестування та профілювання швидкодії дозволяє підбити загальні підсумки розробки та оцінити реальну працездатність інформаційної системи моніторингу TTMON. Створення даного програмного комплексу дозволило повністю вирішити початкову інженерну проблему – автоматизувати наскрізний контроль за життєвим циклом рекламних креативів у межах маркетингової агенції. Результати випробувань у середовищі, максимально наближеному до реального, підтвердили стабільність взаємодії всіх модулів, архітектурну зрілість системи розподілу ролей та високу точність обчислювального аналітичного двигуна. Платформа продемонструвала готовність до експлуатації, забезпечуючи медіабаєрам та керівникам зручний інструментарій для мінімізації фінансових втрат, спричинених деградацією ефективності відеоматеріалів.

Окрему увагу під час підсумкового аналізу результатів впровадження було приділено оцінюванню базового рівня інформаційної безпеки веб-застосунку. Оскільки платформа оперує комерційними та фінансовими даними, захист від поширених мережевих вразливостей був реалізований на рівні системної архітектури за рахунок використання вбудованих інструментів безпеки фреймворку Django.

У системі було повністю нівельовано загрозу підробки міжсайтових запитів завдяки активації механізму перевірки спеціальних криптографічних токенів. Спеціалізований проміжний шар безпеки автоматично генерує унікальний одноразовий токен для кожної активної сесії користувача та впроваджує його у приховані поля всіх HTML-форм, що здійснюють зміну стану бази даних (наприклад, під час створення технічного завдання або зміни доступу до бізнес-центрів). Під час отримання POST-запиту сервер порівнює надісланий маркер із зафіксованим у сесії, що повністю унеможливлює виконання несанкціонованих дій зловмисниками від імені авторизованого співробітника.

Паралельно було забезпечено надійний захист від міжсайтового скриптингу, який полягає у спробах впровадження шкідливого виконуваного коду в інтерфейс системи через текстові поля. Шаблонізатор фреймворку за замовчуванням виконує примусове екранування всіх змінних, що виводяться на екран. Це означає, що якщо користувач спробує ввести шкідливий скрипт у полі опису технічного завдання або текстового хука, система інтерпретує його як звичайний безпечний текст, нейтралізуючи загрозу виконання стороннього коду в браузерах інших співробітників агенції. Додатковий рівень захисту від ін'єкцій SQL-коду забезпечується використанням об'єктно-реляційного відображення, де всі запити до бази даних PostgreSQL проходять обов'язкову параметризацію на рівні драйверів СУБД.

Попри успішне тестування та високу оцінку працездатності поточна версія інформаційної системи TTMON має значний потенціал для подальшої модернізації. Одним з таких напрямків модернізації є впровадження інструментів асинхронного виконання завдань для фонового оновлення фінансової статистики. На поточному етапі обчислення показників відбувається безпосередньо під час HTTP-запиту, коли користувач відкриває сторінку дашборду. У разі масштабування агенції та підключення сотень великих бізнес-центрів, прямі послідовні запити до зовнішніх серверів соціальної мережі призведуть до критичного зростання часу очікування та виникнення помилок тайм-ауту веб-сервера.

Для вирішення цієї інженерної проблеми заплановано інтеграцію розподіленої черги завдань Celery, де як брокер повідомлень буде використано високопродуктивне сховище даних в оперативній пам'яті Redis [17]. Це дозволить винести процеси регулярного опитування зовнішніх API та складні математичні обчислення рівнів вигорання у фонові періодичні процеси, що запускаються автоматично за розкладом. Веб-сервер Django при цьому буде миттєво віддавати користувачу вже готові, попередньо розраховані та збережені в базі даних фінансові метрики, зберігаючи ідеальну швидкодію інтерфейсу.

Третім перспективним вектором розвитку є інтеграція модулів штучного інтелекту для автоматизації процесів проектування креативів. Заплановано

розширення форми створення технічного завдання шляхом підключення зовнішніх мовних моделей через програмні інтерфейси.

Штучний інтелект зможе аналізувати історичні дані успішних рекламних кампаній агенції, виявляти закономірності у текстових хуках, що принесли найбільший ROI, та автоматично генерувати текстові рекомендації для нових ТЗ безпосередньо в момент їхнього заповнення медіабаєром. Це дозволить суттєво підвищити якість початкових маркетингових матеріалів, зменшити відсоток відхилених завдань на етапі перевірки та скоротити витрати часу на рутинне проектування сценаріїв для відеороликів.

Таким чином, розроблена інформаційна система TTMON не лише успішно вирішує актуальні завдання автоматизації моніторингу ефективності маркетингової діяльності на поточному етапі, а й має гнучку, масштабовану архітектуру. Закладений інфраструктурний та програмний потенціал дозволяє легко інтегрувати нові аналітичні інструменти, черги асинхронних завдань та інтелектуальні підсистеми генерації контенту, що забезпечує довгострокову актуальність та високу комерційну цінність програмного комплексу в умовах динамічного розвитку ринку цифрової реклами.

ВИСНОВКИ

Робота над розробкою веб-орієнтованої системи моніторингу ефективності рекламних кампаній у соціальній мережі TikTok дала змогу не просто реалізувати технічне рішення, а фундаментально переосмислити підхід до управління маркетинговими бюджетами та комунікацією всередині агенцій. Початковий аналіз показав, що процеси відстеження результативності відеороликів та постановки технічних завдань дизайнерам значною мірою залежать від ручного контролю, використання розрізнених месенджерів та електронних таблиць. Це не тільки уповільнює реакцію на падіння ефективності реклами, призводячи до фінансових втрат через неконтрольоване «вигорання» креативів, але й створює зайве комунікаційне навантаження на співробітників. Саме тому завдання створити єдину цифрову платформу, яка б автоматизувала аналітику та зробила робочий процес створення рекламних матеріалів прозорішим і структурованішим, виглядало максимально обґрунтованим.

У межах дипломної роботи було реалізовано повноцінну багатокористувацьку систему, архітектура якої враховує специфічні ролі співробітників і сценарії їхньої взаємодії. Впроваджено надійну систему розмежування прав доступу, побудовано власний аналітичний двигун на базі фреймворку Django та підготовлено інтеграційні механізми взаємодії з TikTok Marketing API. Окремим здобутком стала розробка інтерактивного модуля управління технічними завданнями, який підтримує технологію Drag-and-Drop, асинхронне накопичення медіафайлів через DataTransfer API та динамічний попередній перегляд контенту, що перетворює систему на зручний щоденний інструмент для роботи.

Окрему увагу під час розробки було приділено питанням продуктивності, безпеки та стабільності роботи застосунку. Проведене функціональне тестування підтвердило безвідмовну роботу алгоритмів зміни статусів завдань та математичну точність формул обчислення ключових метрик, які успішно пройшли жорстку перевірку на масивах згенерованих тестових даних. Використання оптимізованих запитів до бази даних PostgreSQL дозволило

успішно усунути проблему надлишкових звернень, забезпечивши миттєвий рендеринг динамічних дашбордів навіть за умов масштабування даних. Окрім того, конфігурація захищених тунелів та налаштування віддалених репозиторіїв продемонстрували повну готовність архітектури до переходу в промислову експлуатацію.

Було чітко окреслено подальші вектори розвитку платформи: впровадження асинхронних черг завдань для фонового оновлення статистики, а також інтеграція модулів штучного інтелекту для автоматичної генерації рекомендацій до нових креативів. Платформа TTMON вже зараз автоматизує ті процеси моніторингу та делегування, які раніше потребували залучення кількох людей і значного часу. З урахуванням усього спроєктованого, реалізованого та протестованого можна зробити висновок, що поставлена мета роботи була повністю досягнута. Створена інформаційна система ефективно виконує закладені функції та має потужний потенціал для розвитку, що дозволить агенціям суттєво заощаджувати рекламні бюджети та підвищити загальну результативність маркетингових команд.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Spark Ads 101: Make TikTok videos into ads [Електронний ресурс]. – Режим доступу: https://ads.tiktok.com/business/en-US/blog/spark-ads-101-make-tiktoks-into-ads. – Назва з екрана. – Дата звернення: 20.12.2025.
2. How TikTok recommends videos #ForYou. TikTok Newsroom [Електронний ресурс]. – Режим доступу: https://newsroom.tiktok.com/en-us/how-tiktok-recommends-videos-for-you. – Назва з екрана. – Дата звернення: 12.12.2025.
3. Why TikTok Campaigns Burn Out in 3-5 Days [Електронний ресурс]. – Режим доступу: https://affiliatevalley.com/news/why-tiktok-campaigns-burn-out-in-3-5-days/. – Назва з екрана. – Дата звернення: 26.01.2026.
4. Introduction to TikTok Ads Manager. TikTok Business Help Center [Електронний ресурс]. – Режим доступу: https://ads.tiktok.com/help/article/tiktok-ads-manager-intro?lang=en. – Назва з екрана. – Дата звернення: 19.12.2025.
5. Measurement and attribution. AppsFlyer [Електронний ресурс]. – Режим доступу: https://www.appsflyer.com/products/measurement/. – Назва з екрана. – Дата звернення: 10.12.2025.
6. What is Jira Software? Atlassian [Електронний ресурс]. – Режим доступу: https://www.atlassian.com/software/jira/guides/getting-started/introduction#dig-into-specific-features. – Назва з екрана. – Дата звернення: 26.12.2025.
7. ISO/IEC/IEEE 29148:2018. Systems and software engineering – Life cycle processes – Requirements engineering. ISO [Електронний ресурс]. – Режим доступу: https://www.iso.org/obp/ui/en/#iso:std:iso-iec-ieee:29148:ed-2:v1:en. – Назва з екрана. – Дата звернення: 29.12.2025.
8. Django documentation. Design philosophies [Електронний ресурс]. – Режим доступу: https://docs.djangoproject.com/en/stable/misc/design-philosophies/. – Назва з екрана. – Дата звернення: 07.02.2026.

9. PostgreSQL 16.0 Documentation [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/16/index.html. – Назва з екрана. – Дата звернення: 08.01.2026.
10. User authentication in Django. Django Documentation [Електронний ресурс]. – Режим доступу: https://docs.djangoproject.com/en/stable/topics/auth/. – Назва з екрана. – Дата звернення: 15.02.2026.
11. Authentication. TikTok Business API [Електронний ресурс]. – Режим доступу: https://business-api.tiktok.com/portal/docs?id=1738373164380162. – Назва з екрана. – Дата звернення: 22.02.2026.
12. The OAuth 2.0 Authorization Framework. IETF [Електронний ресурс]. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc6749. – Назва з екрана. – Дата звернення: 21.02.2026.
13. Vincent W. S. Django for Professionals: Production-Ready Web Development / W. S. Vincent. – 2022. – 293 с.
14. Feldroy D. Two Scoops of Django 3.x: Best Practices for the Django Web Framework / D. Feldroy, A. Feldroy. – Two Scoops Press, 2020. – 485 с.
15. DataTransfer – Web APIs | MDN. Mozilla Developer Network [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API/DataTransfer. – Назва з екрана. – Дата звернення: 15.04.2026.
16. Myers G. J. The Art of Software Testing / G. J. Myers, C. Sandler, T. Badgett. – John Wiley & Sons, 2012. – 254 с.
17. Celery: Distributed Task Queue [Електронний ресурс]. – Режим доступу: https://docs.celeryq.dev. – Назва з екрана. – Дата звернення: 13.05.2026.

ДОДАТКИ

Додаток А

Календарний план

№ з/п	Назва етапів виконання бакалаврської кваліфікаційної роботи	Строк виконання етапів бакалаврської кваліфікаційної роботи	Примітки
1	Одержання завдання на бакалаврську роботу, визначення структури та розробка календарного плану.	01.12.2025 – 05.12.2025	Затверджений календарний план
2	Дослідження предметної області маркетингової діяльності та аналіз існуючих засобів автоматизації моніторингу реклами.	06.12.2025 – 15.12.2025	Огляд літературних джерел аналізу
3	Опрацювання технічної документації TikTok API та порівняльний аналіз інструментів для розробки .	16.12.2025 – 24.12.2025	Обґрунтований технологічний стек
4	Формулювання функціональних вимог до системи та моделювання бізнес-процесів взаємодії користувачів у нотації BPMN.	25.12.2025 – 03.01.2026	Специфікація вимог, діаграми BPMN
5	Проєктування концептуальної та логічної структури бази даних, побудова діаграми сутність-зв'язок.	04.01.2026 – 12.01.2026	Схема БД, ER-діаграма у записці
6	Розробка загальної архітектури інформаційної системи та побудова діаграм прецедентів використання UML.	13.01.2026 – 20.01.2026	UML-діаграми архітектури
7	Математичне моделювання та формалізація алгоритму визначення коефіцієнта вигорання рекламних креативів	21.01.2026 – 28.01.2026	Математична модель та формули
8	Оформлення та редагування першого (теоретичного) та другого (проєктного) розділів пояснювальної записки.	29.01.2026 – 05.02.2026	Текст Розділів 1 та 2
9	Ініціалізація проєкту Django, налаштування віртуального оточення та конфігурація приватного репозиторію на GitHub.	06.02.2026 – 12.02.2026	Робочий репозиторій, початкова кодова база
10	Програмна реалізація модуля управління користувачами та інтерфейсів безпеки зміни паролів.	13.02.2026 – 19.02.2026	Програмний код модуля users
11	Розробка підсистеми інтеграції: реалізація протоколу авторизації	20.02.2026 – 27.02.2026	Авторизація через TikTok

	OAuth 2.0 та обробників зворотного виклику.		Sandbox
12	Створення моделей даних для обліку рекламних кампаній, кабінетів та креативів у додатку analytics.	28.02.2026 – 06.03.2026	Таблиці СУБД для аналітики
13	Реалізація аналітичного двигуна обчислення фінансових та маркетингових показників через декоратори @property.	07.03.2026 – 14.03.2026	Алгоритми обчислення метрик у коді
14	Програмна реалізація модуля технічних завдань та розробка кінцевого автомата управління життєвим циклом ТЗ.	15.03.2026 – 22.03.2026	Робочий процес створення та перевірки ТЗ
15	Проєктування шаблонів та адаптивна верстка користувацьких інтерфейсів системи на основі фреймворку Bootstrap.	23.03.2026 – 29.03.2026	Набір базових HTML/CSS шаблонів
16	Розробка динамічних елементів інтерактивного дашборду: підключення календаря Flatpickr та логіки мультифільтрації.	30.03.2026 – 05.04.2026	Інтерфейс фільтрації статистики
17	Реалізація клієнтського модуля кастомізації аналітичної таблиці за допомогою Sortable.js із збереженням стану в localStorage.	06.04.2026 – 12.04.2026	Динамічне перетягування колонок таблиці
18	Програмна реалізація асинхронного накопичення та валідації медіафайлів у формі створення ТЗ через DataTransfer API.	13.04.2026 – 19.04.2026	Сценарії завантаження кількох референсів
19	Інтеграція відеоплеєра та інструментів миттєвого попереднього перегляду готових відеороликів через Blob URL.	20.04.2026 – 25.04.2026	Інтерактивна картка ТЗ з медіаплеєром
20	Інфраструктурне налаштування мережевого шлюзу утиліти ngrok для проксіювання HTTPS-запитів на локальний сервер розробника.	26.04.2026 – 29.04.2026	Конфігурація безпечного тунелю для вебхуків
21	Виконання сценаріїв функціонального тестування підсистеми автентифікації та розмежування доступу користувачів.	30.04.2026 – 03.05.2026	Складання таблиць тестування безпеки
22	Наповнення СУБД сирими мокованими даними для перевірки точності агрегації статистики та тригерів вигорання креативів.	04.05.2026 – 06.05.2026	Верифікація обчислювальних формул двигуна
23	Профілювання продуктивності інтерфейсів, виявлення проблеми N+1 запитів та оптимізація через	07.05.2026 – 09.05.2026	Зниження навантаження на PostgreSQL

	select_related і prefetch_related.		
24	Розгортання статичного презентаційного шару на GitHub Pages та конфігурація публічних DNS-записів домену ttmon.site.	10.05.2026 – 12.05.2026	Активний домен та перша вебсторінка системи
25	Оформлення третього та четвертого розділів пояснювальної записки.	13.05.2026 – 14.05.2026	Повний текст основної частини роботи
26	Формування вступу, загальних висновків до роботи, переліку умовних позначень та списку використаних джерел.	15.05.2026 – 16.05.2026	Фінальна структура пояснювальної записки
27	Перевірка тексту на наявність плагіату, остаточне нормоконтрольне редагування та підготовка демонстраційних слайдів до захисту.	17.05.2026 – 22.05.2026	Готовий проєкт бакалаврської роботи

Студент _____

(підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи_____

(підпис) (ПІБ)

Додаток Б

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій Декларація про академічну доброчесність та використання ШІ

Я, Безушко Тарас Олегович, здобувач НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота на тему «Веб-орієнтована система для моніторингу ефективності рекламних компаній в соціальній мережі TikTok» є результатом моєї особистої праці.

2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.

3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):

інструменти ШІ Gemini, DeepL були використані для:

- перекладу іншомовних джерел;
- стилістичного редагування та перевірки граматики;
- допомоги у написанні/відлагодженні програмного коду;
- інше: пошуку додаткових інструментів у розробці програмного продукту.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«22» Травня 2026 р. _____ **Тарас БЕЗУШКО**