КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра математичної інформатики

«До захисту допущено»
Завідувач кафедри
В. М. Терещенко _____
(підпис)

«___» _____20__ р.

Кваліфікаційна робота
на здобуття ступеня бакалавра
за спеціальністю 122 Комп'ютерні науки
на тему:

РОЗРОБКА ВЕБ-ПЛАТФОРМИ ДЛЯ СТВОРЕННЯ РЕКЛАМНИХ ІНТЕГРАЦІЙ У ВІДЕОМАТЕРІАЛІ

Виконав студент 4 курсу
Леонов Олексій Андрійович _____
(підпис)

Науковий керівник:
доцент, кандидат фізико-математичних наук
Панченко Тарас Володимирович _____
(підпис)

Засвідчую, що в цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2022

РЕФЕРАТ

Обсяг роботи 43 сторінки, 13 ілюстрацій, 30 джерел посилань.

ХМАРНІ ТЕХНОЛОГІЇ, ВЕБ-ДОДАТОК, ОБРОБКА ВІДЕО-ФАЙЛІВ, РОЗПІЗНАВАННЯ СЦЕН У ВІДЕО, РОЗРОБКА ВЕБ-СЕРВІСУ, РОЗГОРТАННЯ ДОДАТКУ, ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОДУКТУ, ПЛАТФОРМА ДЛЯ СТВОРЕННЯ РЕКЛАМНИХ ІНТЕГРАЦІЙ.

Об’єктом роботи є процес розробки програмного забезпечення для укладання рекламних договорів між рекламодавцем та авторами відеоконтенту разом із автоматичною інтеграцію реклами у відео для зменшення витрат часового ресурсу в обох сторін.

Метою роботи є створення прототипу веб-додатку, що може виконувати функції укладання рекламних договорів та автоматичної інтеграції відео-реклами у відеоролики.

Результати роботи: виконано аналіз поточного ринку цифрової реклами, виявлено проблеми, що не мають стандартизованого рішення, але можуть бути автоматизовані задля підвищення ефективності роботи, проведено аналіз разом із наведенням переваг та недоліків способів збереження відеофайлів, інструментів аналізу відео для виявлення в ньому сцен, інструментів для автоматизованого редагування відео, розроблено прототип веб-додатку.

Розроблений програмний продукт може бути використаний у бізнесі на ринку цифрової реклами після подальшої роботи на прототипом та вдосконалення його як з продуктової точки зору, так і з технічної.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА ВИЗНАЧЕНЬ

СУБД – Система Управління Базами Даних

API – Application Programming Interface

AWS – Amazon Web Services

CLI – Command Line Interface

HDFS – Hadoop Distributed File System

HTTP – Hypertext Transfer Protocol

JSON – JavaScript Object Notation

REST – Representational State Transfer

UI – User Interface

ВСТУП

Оцінка сучасного стану об’єкта розробки. Цифрова реклама стала одним з найбільш незамінних маркетингових інструментів у всьому світі. Завдяки зростанню рівня проникнення Інтернету та постійно зростаючому попиту на онлайн-контент, особливо під час пандемії, глобальні витрати на цифрову рекламу, за прогнозами, можуть досягти рекордного рівня в 393 мільярдів доларів США у 2022 році [1]. Більше того, в 2021 році загальна кількість витрат на рекламу зросла на 19.9% замість 15%, як було передбачено раніше, і в той самий час більшість з цих нових витрат на рекламу йде на цифрову рекламу, що зросла на 29.1% замість 20.4% [2]. Очевидно, що інтернет-реклама з кожним роком стає все більш популярною для бізнесу.

Згідно з нещодавнім дослідженням HubSpot [3], 85% компаній використовують відеорекламу як частину своїх маркетингових стратегій. З цих 85% 92% кажуть, що це важлива частина їхньої стратегії, і ця цифра постійно зростає протягом останніх п’яти років.

Актуальність роботи та підстави для її виконання. Згідно з вище наведеною статистикою, можна зробити висновок, що ринок цифрової реклами зростає дуже стрімко і інтернет-реклама є дуже затребуваною серед бізнесів. Одним з найпоширеніших видів інтернет-реклами є відеореклама, що в основному доставляється до споживача за допомогою інтеграцій у відеоконтент авторів на різних відео-платформах – YouTube, Instagram, TikTok та інші.

Існують такі способи інтеграції відео-реклами на платформах:

1. безпосередньо купівля рекламної кампанії на платформі;
2. інтеграція реклами як частини відеоролика за допомогою укладання договору між рекламодавцем та виконавцем.

Перший спосіб, як правило, є найпростішим. Все, що для нього потрібно, це мати саму відеорекламу для рекламної кампанії та знання, для якої аудиторії вона має бути призначена. Рекламодавець власноруч налаштовує параметри рекламної

кампанії на платформі, і в результаті відеореклама інтегрується в певних місцях ролику. В цього способу є той недолік, що, згідно зі статистикою, 43% всіх інтернет-користувачів використовують програмне забезпечення, що блокує інтернет-рекламу у браузері [4]. Більше того, з'являються можливості купівлі спеціальних підписок на платформах, як наприклад, Youtube Music, що прибирають рекламу, яка інтегрується самою відео-платформою.

Другий спосіб потребує більшої залученості рекламодавця. Спосіб полягає в тому, щоб самотужки знайти виконавця, що погодиться інтегрувати рекламу у свою роботу. На даний момент не існує єдиного рішення або стандарту, який міг би звести зайві зусилля обох сторін у рекламному договорі до мінімума.

Сам стрімкий розвиток та зростання ринку відео-реклами із року в рік очевидно доводить, що тема створення нових способів рекламних інтеграцій є актуальною.

Мета й завдання роботи. Метою даної дипломної роботи є створення прототипу нової веб-платформи, на якій надається можливість створення рекламних відео-інтеграцій між рекламодавцем та виконавцем. Для цього було поставлено наступні завдання:

1. дослідити існуючі рішення, що надають надають подібні послуги;
2. скласти продуктові та технічні вимоги до платформи;
3. реалізувати прототип та провести його тестування.

Об'єкт, методи й засоби розроблення. Об'єктом розроблення є веб-застосунок, що надає можливість створення рекламних відео-інтеграцій.

Розробка веб-застосунку була проведена за допомогою розбиття рішення на функціональні модулі, що були окремо розроблені та протестовані з урахуванням залежності одного модуля від іншого.

Для розробки прототипу було використано мову програмування JavaScript та середовище виконання NodeJS як для серверної частини застосунку, так і для клієнтської, оскільки ця мова програмування надає можливості легко та швидко розробити прототип програми, а потім ітеративно його вдосконалювати. Для розробки візуальної частини (UI) було використано движок шаблонів Pug. Для

зберігання та роботи з даними була використана документо-орієнтована СУБД MongoDB, а також сервіси платформи хмарних обчислень AWS такі як S3, Elastic Transcoder та Rekognition.

Можливості сфери застосування. Розроблений веб-додаток може використовуватись рекламодавцями, що шукають можливості інтеграцій відеореклами на інтернет-платформах, а також самими авторами відеоконтенту для спрощення процесу створення та виконання рекламних договорів.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ

1.1. Виокремлення проблематики

Одним із способів інтеграції реклами у відеоконтент є самостійний пошук автора, що створює та публікує свої відео-роботи на спеціальних платформах таких як YouTube, та створення з ним рекламного договору тим чи іншим способом. Даний спосіб має такі переваги над безпосередньо купівлею рекламної кампанії на платформі:

1. рекламодавець самостійно контролює у якого автора, в якому відео та на якому моменті буде інтегрована реклама;
2. контроль над типом аудиторії, для якої буде показано рекламу;
3. більша гарантія того, що реклама дійде до споживача, не дивлячись на те, що у користувача може бути встановлене програмне забезпечення, що блокує рекламні вставки на відео-платформі;
4. більша довіра до рекламодавця з боку споживача реклами, що є постійним глядачем автора, що виконує рекламний договір.

Однак у даного способу є недоліки. Рекламодавцю необхідно витрачати більше робочого часу на те, щоб створити цю рекламну кампанію: необхідно знайти самого виконавця за визначеними критеріями, знайти з ним спосіб зв’язку, запропонувати співпрацю, знайти спільний формат роботи, підписати договір, проконтролювати виконання, провести оплату.

Перелічені кроки вище не мають жодної стандартизації або єдиного рішення, що призводить до того, що робота виконується не так ефективно, як це було б можливо за його наявності.

1.2. Постановка задачі

Для вирішення раніше згаданої проблематики пропонується створити новий веб-платформу, яка могла би вирішити проблему відсутності єдиного рішення для створення рекламних інтеграцій безпосередньо у відео виконавців.

Веб-платформа має виконувати наступні функції:

1. реєстрація користувача як виконавця або рекламодавця;
2. завантаження відео: контенту та реклами;
3. можливість пошуку відеоматеріалу, що цікавить рекламодавця;
4. створення рекламних договорів між замовником та виконавцем;
5. інтеграція відеореклами у відеоконтент;
6. монетизація: створення внутрішнього гаманця для створення більшої довіри до платформи.

Для етапу інтеграції реклами у відеоконтент необхідно створити зручний інтерфейс для виконавця, що надасть змогу автоматично виконувати вставку реклами на заданій мітці часу. Більше того, необхідно зробити так, щоб користувачу платформа самостійно пропонувала підходящі часові мітки, де реклама може бути інтегрована у відео, а користувач може обрати один із запропонованих варіантів, або ж задати власний варіант.

1.3. Потенційні конкуренти

Ринок інтернет-реклами має дуже стрімке зростання, і можна стверджувати, що його причиною є те, що в ньому задіяні такі технічні гіганти як Google, Facebook, Netflix, Amazon, HubSpot та інші. Логічно, що стрімке зростання галузі інтернет-реклами ще більше привертає увагу тих самих гігантів. Це призводить до того, що якість продуктів у цій сфері стає все більш високою і, більше того, компанії прагнуть стати монополістами на цьому ринку.

Найближчим конкурентом до запропонованого у цій роботі продукту з’явився нещодавно у 2022 році. Цим конкурентом є TikTok: Branded Mission [26].

Ідея даного продукту полягає в тому, щоб на платформі соціальної мережі TikTok дозволити рекламодавцям створювати конкурси на рекламну кампанію для власного продукту. Оголошується рекламний бюджет і анонсуються вимоги до короткого рекламного ролику, який має бути записаний виконавцем. Найкраща робота обирається рекламодавцем і автор рекламного відеоролику отримує за роботу грошову винагороду.

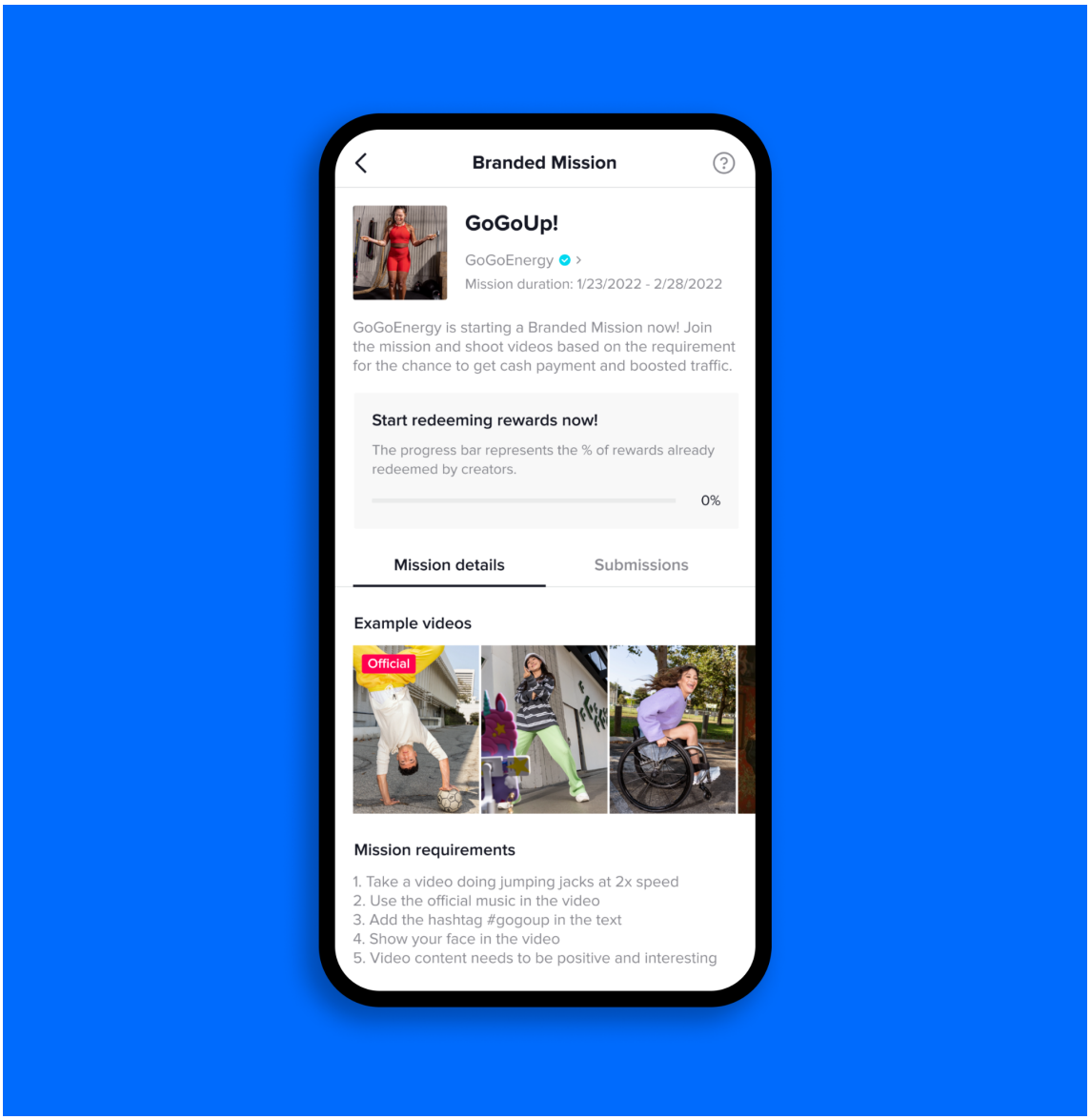

Рисунок 1 – Інтерфейс TikTok: Branded Mission [26]

РОЗДІЛ 2. РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ ДОДАТКУ

2.1. Зберігання відео

Задача зберігання великих файлів на диску не є тривіальною задачею як може здатись на перший погляд. Для вирішення цієї проблеми декому може спасти на думку, що достатньо створити деяку таблицю в такій СУБД як, наприклад, MySQL і просто зберігати файл у вигляді бінарних даних прямо в одній з колонок таблиці. Як виявляється, дане рішення не підходить для задачі зберігання відео файлів.

По-перше, зберігання великих файлів у базі даних збільшує час запитів до неї. Оскільки бази даних використовують оперативну пам’ять для пришвидшення запитів, зберігання в ній великих файлів призведе до того, що ці самі файли будуть зберігатись в оперативній пам’яті, що є досить обмеженим ресурсом серверів. Іншими словами, коли база даних зайнята тим, щоб повернути великий файл на отриманий запит, інші запити не можуть бути виконані, оскільки на це немає ресурсів.

По-друге, підтримка бази даних, що спрямована на зберігання великих файлів, стає набагато складнішою. Велику базу даних підтримувати набагато складніше, ніж маленьку. Створення резервних копій, відтворення бази даних за допомогою резервних копій, створення індексів та інші задачі стають набагато складнішими.

По-третє, для того, щоб зберегти файл у базі даних, його необхідно конвертувати у тип даних, що підтримується нею. Наприклад, якщо обрати збереження файлу у вигляді тексту за допомогою конвертації файлу у формат base64, як мінімум буде необхідно написати додаткову логіку на сервері з конвертації файлу, що буде займати зайві ресурси. Більше того, файли, що зберігаються у даному форматі, займають на 33% більше місця ніж початковий файл [5].

Отже, можна зробити висновок, що база даних не є підходящим варіантом для задачі збереження великих файлів таких як відеофайлів.

Для цієї задачі набагато краще підходить варіант зберігання файлів у файловій системі. Слід зауважити, що можна використати звичайну файлову систему, що є в наявності на сервері, однак ще кращим варіантом буде використання розподіленої файлової системи. Одним з прикладів такої файлової системи є HDFS [6].

HDFS – це розподілена, масштабована та портативна файлова система, написана мовою програмування Java для фреймворку Hadoop. Це така файлова система, що, як виходить із назви, замість того, щоб працювати на одному єдиному сервері, розподілена між декількома машинами, що надає дуже високу загальну пропускну здатність загалом.

HDFS розроблено для надійного зберігання дуже великих файлів на машинах у великому кластері. Він зберігає кожен файл у вигляді послідовності блоків; усі блоки у файлі, крім останнього, мають однаковий розмір. Блоки файлу реплікуються для відмовостійкості. Розмір блоку та коефіцієнт реплікації налаштовуються для кожного файлу.

Ще одним варіантом для вирішення задачі зберігання відеофайлів є використання хмарного сервісу AWS S3 [7].

AWS S3 – Amazon Simple Storage Service, хмарний сервіс для зберігання об’єктів, що пропонує провідну в галузі масштабованість, доступність даних, безпеку та продуктивність [7]. Як і для всіх сервісів Amazon, для цього сервісу надається зручний SDK для популярних мов програмування, в тому числі для JavaScript.

Основною одиницею зберігання в Amazon S3 є об’єкт, що містить файл із пов’язаним ідентифікатором та метаданими. Ці об’єкти зберігаються в бакетах (від англ. bucket), що функціонують так само, які і директорії у файловій системі, і які розташовуються у регіоні AWS за вибором.

По суті цей сервіс схожий на NoSQL базу даних. Кожен бакет це нова база даних, у якої ключем доступу є шлях до директорії, а значенням є бінарний об’єкт (файл).

Однак не варто вважати, що AWS S3 це розподілена файлова система. Це сховище об’єктів AWS, а не файлова система, тоді як HDFS — це розподілена файлова система, призначена для зберігання великих даних, де гарантована відмовостійкість. S3 – це сховище об’єктів, тобто всі дані в S3 зберігаються як об’єкти з пов’язаним з ним ключем (ім’ям документа). S3 насправді є нескінченним сховищем у хмарі, але HDFS – ні. HDFS розміщується на фізичних машинах, тому ви можете виконувати там будь-яку програму. Ви не можете нічого виконувати на S3, оскільки це лише сховище об’єктів, а не файлова система.

Для вирішення задачі зберігання відео було обрано останній варіант – AWS S3. Таке рішення було прийняте із-за наступних причин:

1. **Еластичність сервісу.** Досить важко передбачити скільки ресурсів буде необхідно для того чи іншого сервісу, в той час як S3 дозволяє зберігати файли без думок про кількість наявних ресурсів, бо по суті немає ніяких обмежень на їх використання;
2. **Доступність і довговічність.** Amazon заявляє про 99.999999999% довговічність та 99.99% доступність, що є більшим, ніж пропонує більшість внутрішніх сервісів різних організацій. В той самий час важко підрахувати ці показники для HDFS, однак для більшості кластерів доступність є меншою, ніж 99.9%, що є по суті як мінімум 9 годин часу простою на рік [8].
3. **Доступність використання.** AWS S3 є простим сервісом, особливо у порівнянні із HDFS. Для того, щоб почати використовувати не треба робити складні налаштування кластеру, все працює так само, як звичайна NoSQL база даних, в якій ключем є шлях до файлу, а значенням – сам файл.

Давайте розглянемо яким чином може відбуватись процес завантаження відео на платформу.

Запит на завантаження відео буде виконуватись з клієнта за допомогою розробленого нами REST API, розробка якого буде детально досліджена та описана пізніше в цій роботі. Оскільки S3 – сервіс, що нагадує NoSQL базу даних, напрошується очевидний спосіб завантаження відео у сховище: клієнт виконує HTTP-запит на сервер з бінарним файлом у тілі запиту, а в той час сервер той самий бінарний файл завантажує як об’єкт у хмару S3.

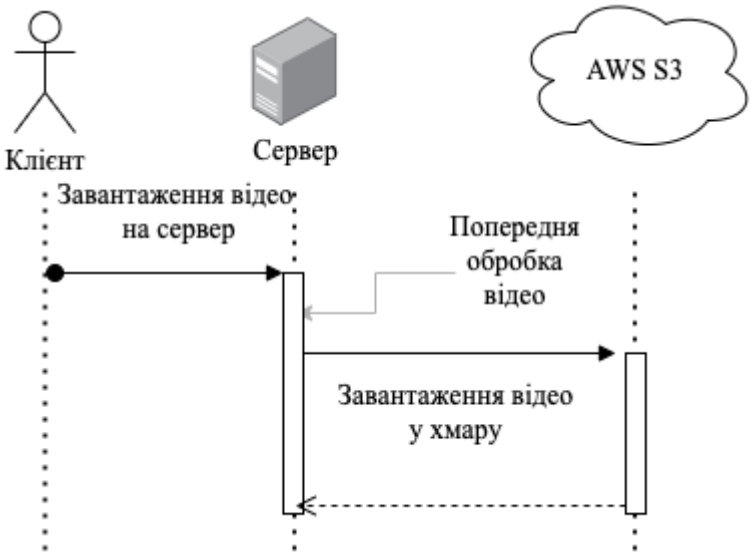

Рисунок 2 – Завантаження відео на сервер

У цього способу є певний недолік – сервер виконує зайву роботу у вигляді попередньої обробки відео перед його завантаженням на хмару. Тобто спочатку сервер повинен зберегти файл, що передає клієнт у запиті, собі на диск, а потім завантажити цей файл на хмару.

Для уникнення зайвої роботи, що виконує сервер у даному способі завантаження відео, було б зручно, якби клієнт міг напряму завантажувати відео-файл на хмару. І, на щастя, такий спосіб існує. AWS S3 надає можливість створення спеціального посилання, за допомогою якого клієнт згодом може завантажити будь-який об’єкт на сервіс напряму. У цьому посиланні надаються спеціальні метадані такі як підпис, алгоритм підрахунку підпису та строк дії посилання для завантаження для того, щоб зберегти безпечність завантаження об’єкту.

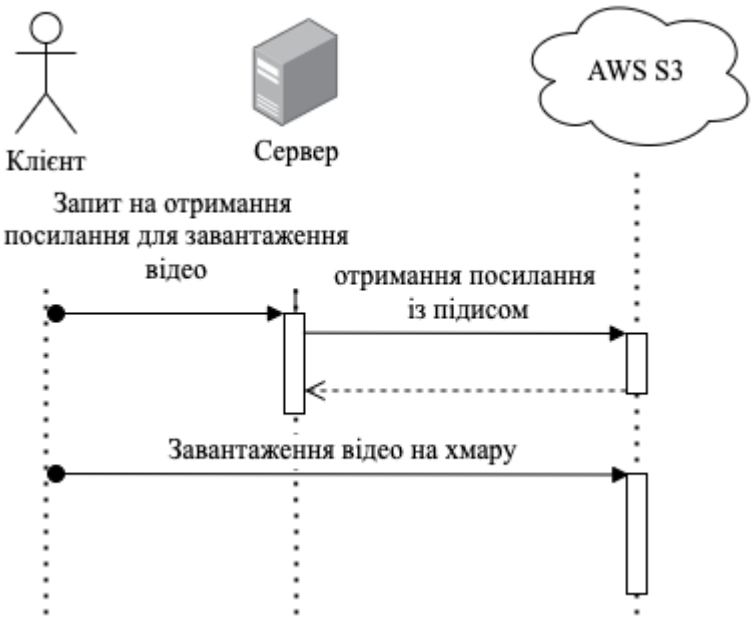

Рисунок 3 – Завантаження відео напряму на хмару

У даному способі зникає зайва попередня обробка відео-файлу для завантаження відео на хмару.

2.2. Аналіз відео для виявлення потенційних місць рекламних інтеграцій

Для спрощення та пришвидшення створення рекламного договору було вирішено додати на платформу систему рекомендацій часових кодів у відео, куди може бути інтегрована реклама. Нижче розглянемо способи та інструменти, за допомогою яких це можна реалізувати.

Логічно, що реклама не має переривати відео на моменті, де людина говорить, або просто посередині якоїсь сцени. Це значно зменшує якість контенту та призводить до невдоволення глядачів переглядом. Для нашого застосунку було вирішено розробити логіку рекомендацій місць для інтеграцій реклами на основі виявлення сцен у відео. Для того, щоб реклама була природно інтегрована у відео, її варто вставляти між сценами. Іншим варіантом є вставка реклами у технічних паузах, як, наприклад, чорний екран, технічний шум, або ж відео-перебивка.

Отже, для виконання цієї частини завдання нам знадобиться навчитися виявляти сцени у відео. Тож давайте розглянемо наявні інструменти, що підходять для цієї задачі.

Одним з можливих інструментів для цієї задачі є CLI додаток, що називається PySceneDetect [9].

PySceneDetect – CLI додаток та Python бібліотека, що призначена для виявлення змін сцен у відео та автоматичного його розбиття на менші шматки відео, що відповідають виявленим сценам. Це безкоштовне та open-source програмне забезпечення, що має декілька можливих способів виявлення сцен. Давайте розглянемо алгоритми, що використовує ця бібліотека для виявлення сцен[10].

Перший доступний метод виявлення сцен називається content-aware detection. Він виявляє так звані джамп-кати (від англ. – jump cut) у відео. Цей алгоритм знаходить місця, де різниця між двома послідовними кадрами перевищує певний поріг, що був встановлений. Зсередини, відбувається конвертація кожного кадру з кодування RGB [11] до HSV [12]. Згодом береться середня різниця між всіма каналами (або опціонально тільки value-канал) від кадру до кадру. Якщо ця різниця перевищує певний поріг, значить була виявлена зміна сцени.

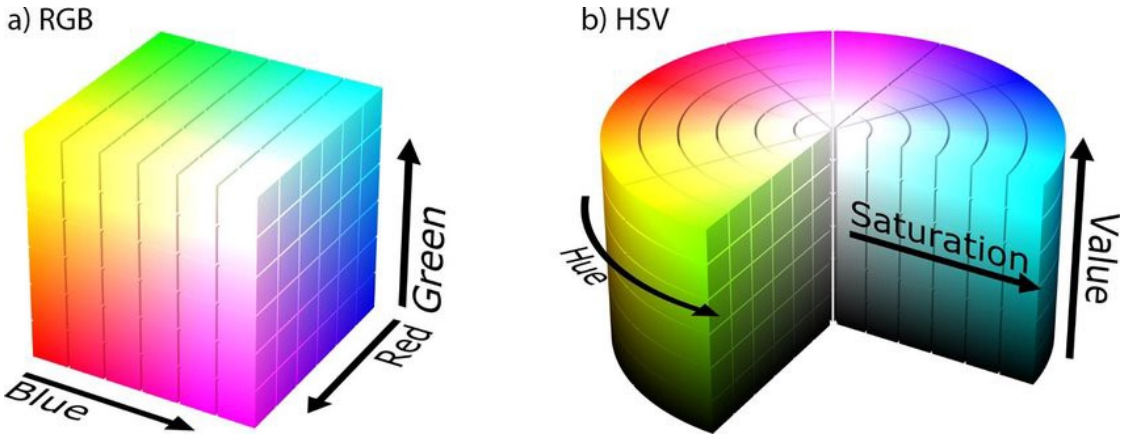

Рисунок 4 – Візуалізація принципу подання кольору за допомогою RGB та HSV [25]

Другий метод виявлення сцен називається adaptive content detection. Його суть полягає в тому, що він подібно до попереднього методу порівнює два суміжні

кадри, але для порівняння використовується рухома середня різниця. Це дозволяє уникати випадків, коли виявляються хибні сцени через швидкий рух камери.

Третій алгоритм називається threshold detection. Це метод, який традиційно використовується в інструментах виявлення сцен. Задля виявлення сцен виконується порівняння інтенсивності або ж яскравості поточного кадру з певним порогом.

Більше того, PySceneDetect дозволяє створювати власні методи виявлення сцен, що згодом можуть бути використані при застосуванні цієї бібліотеки.

Інший інструмент, що може бути використаний для виявлення сцен у відео, це ще один сервіс AWS – Amazon Rekognition Video [13]. Даний сервіс надає широкий спектр аналізу відео та фотографій. За допомогою нього можливо виявляти текст на відео, пошук знаменитостей, виявлення облич та, що більше всього нас цікавить, виявлення сцен.

На жаль, Amazon Rekogntion Video – не open-source продукт і в цій роботі не вдалося знайти конкретні методи, за допомогою яких виконується пошук сцен у відео. Однак можна зробити припущення, що використовується один із вище описаних методів, що використовується у PySceneDetect.

Давайте ближче поглянемо на Amazon Rekognition Segments API, що відповідає за виявлення сцен у відео [14]. Даний сервіс вміє виявляти наступні види сегментів відео: чорні кадри, кольорові смуги, відкриваючі та закриваючі титри, сцени. Більше того, у документації сказано, що даний API може використовуватись для виявлення місць вставки реклами, наприклад, в місцях чорного екрану.

Більше того, даний API є зручним рішенням в нашому випадку, оскільки раніше ми вирішили, що будемо використовувати хмарний сервіс AWS S3 для зберігання відео. Segments API працює виключно з відео, що були завантажені на цей сервіс.

Для початку аналізу відео необхідно викликати даний API разом із такими даними: бакет, в якому знаходиться відео, яке необхідно проаналізувати, та ключ (шлях) до відео. В результаті цього виклику запускається асинхронний процес

аналізу відео. Результатом роботи є об’єкт, в якому надається список сегментів відео (часові коди початку, кінця, інша метаінформація), що були виявлені, разом з інформацією, що це за тип сегменту (технічна перерва або ж сцена) та рівнем впевненості аналізу з дійсним значенням від 0 до 100.

Приклад результату виконання аналізу відео на визначення у ньому сцен у вигляді JSON-об’єкту:


```
{
  "SelectedSegmentTypes": [
    {
      "ModelVersion": "2.0",
      "Type": "SHOT"
    },
    {
      "ModelVersion": "2.0",
      "Type": "TECHNICAL_CUE"
    }
  ],
  "Segments": [
    {
      "DurationFrames": 299,
      "DurationSMPTE": "00:00:09;29",
      "StartFrameNumber": 0,
      "EndFrameNumber": 299,
      "EndTimecodeSMPTE": "00:00:09;29",
      "EndTimestampMillis": 9976,
      "StartTimestampMillis": 0,
      "DurationMillis": 9976,
      "StartTimecodeSMPTE": "00:00:00;00",
      "Type": "TECHNICAL_CUE",
      "TechnicalCueSegment": {
        "Confidence": 90.45006561279297,
        "Type": "BlackFrames"
      }
    },
    …
  ],
  "JobStatus": "SUCCEEDED",
```

```
    "VideoMetadata": [
      {
        "Format": "QuickTime / MOV",
        "FrameRate": 29.970029830932617,
        "Codec": "h264",
        "DurationMillis": 15015,
        "FrameHeight": 1080,
        "FrameWidth": 1920,
        "ColorRange": "LIMITED"

      }
    ],
    "AudioMetadata": [
      {
        "Codec": "aac",
        "DurationMillis": 15015,
        "SampleRate": 44100,
        "NumberOfChannels": 2
      }
    ]
}
```


Недоліком даного API є те, що він не є платним, як і всі інші продукті Amazon. Однак, даний сервіс надає досить щедру тривалість безкоштовного користування.

Проаналізувавши два вищезгадані інструменти для виявлення сцен у відео, було обрано AWS Rekognition Video за наступних причин:

1. Легке користування API, оскільки він працює на основі хмарного сервісу S3, який ми використовуємо для зберігання відео;
2. У випадку обрання PySceneDetect, було б необхідно виконувати додаткову роботу на сервері, що займала б ресурси, або ж взагалі створювати окремий сервіс, що мав би запускатись на окремому сервері, що потребує реалізації комунікації між сервісами у внутрішній мережі. Замість того, щоб витрачати час та ресурси на власний сервіс, варто використати вже існуючий, що, скоріше за все, буде перевищувати в швидкості та якості.

Надалі дану інформацію про сегменти відео можна використовувати для подальшого визначення часових кодів, де оптимально інтегрувати рекламу у відео. Наприклад, можна використати правило пропонувати часові коди після відео сегментів, що за тривалістю перевищують 30 секунд, або ж по середині технічної паузи.

2.3. Автоматична інтеграція реклами у відео

Наступний функціонал, який має бути реалізований у даній роботі, це автоматична вставка реклами у відео. Виконавець, коли погоджується на пропозицію рекламної інтеграції у власне відео, має мати змогу після обрання часового коду, де має бути вставлена реклама, автоматизовано виконати редагування свого відео та вставити рекламу на відповідний часовий код.

Для виконання цієї задачі можна використати існуючий Amazon сервіс, що має назву Amazon Elastic Transcoder [15]. Його використання є досить простим. Так само, як і AWS Rekognition Video, цей сервіс працює з відео, що зберігаються на хмарному сервісі AWS S3. Даний сервіс надає можливість “зшивання” відео сегментів. На вхід він приймає список сегментів, які потрібно “зшити” разом.

Нехай у нас є відеоряд з такими параметрами: *довжина відео = LV*, *часовий код для вставки реклами = AT,* причому $0 \leq AT \leq LV$, та *довжина реклами = LA*. Отже, для того, щоб виконати вставку рекламу на заданому часовому коді у відео, необхідно надати наступні сегменти відео на вхід сервісу в заданому порядку: сегмент відеоряду *[0, AT],* сегмент реклами *[0, LA], сегмент відеоряду [AT, LV]*.

На жаль, даний сервіс є досить дорогим у використанні. На момент написання роботи для безкоштовного використання сервісу надається всього 20 хвилин загального процесингу відео, чого вкрай недостатньо для нашої роботи.

Розглянемо інший інструмент для виконання даної задачі. Найпопулярнішим CLI інструментом для виконання операцій над відео є FFmpeg. FFmpeg є провідним мультимедійним фреймворком, здатний декодувати, кодувати, перекодувати, мультиплікувати, демультиплексувати, фільтрувати та відтворювати

майже все, що створили люди та машини. Він підтримує найнезрозуміліші стародавні формати аж до найсучасніших [16].

Для вставки реклами у відео на потрібному часовому коді необхідно виконати дії, подібні до дій, що ми описали вище для Amazon Elastic Transcoder. Тобто для початку необхідно “розрізати” відео, в яке буде інтегрована реклама, на два інших відео на моменті, де ми плануємо вставити рекламу. А згодом “зшити” три відео в одне єдине, що і буде результатом інтеграції реклами у відео.

Процедура “розрізання” відео за допомогою інструмента FFmpeg виконується досить просто. Нижче наведено приклад виконання даної процедури у командному рядку.

$ ffmpeg -i input.mp4 -ss 00:05:20 -t 00:10:00 -c:v copy -c:a copy output1.mp4

Параметр *-i input.mp4* вказує шлях до вхідного відеофайлу; *-ss 00:05:20* – початок сегменту, який нас цікавить; *-t 00:10:00* – тривалість сегменту в 10 хвилин; *-c:v copy copy -c:a copy* вказують на те, що відео та аудіо потрібно залишити в тому самому кодуванні, що й у вхідному відео; *output1.mp4* – шлях до результуючого відео.

Для того, щоб виконати “розрізання” відео на певному часовому коді (T), необхідно виконати наступні 2 команди:

$ ffmpeg -i input.mp4 -ss 00:00:00 -t T -c:v copy -c:a copy output1.mp4
$ ffmpeg -i input.mp4 -ss T -c:v copy -c:a copy output2.mp4

Наступною задачею, що треба виконати для реалізації інтеграції реклами у відео, є “склеювання” відео. Відеофайли є не простими за структурою файлами. Зміст таких файлів залежить від кодування відео, і кожен вид кодування відео має свої переваги та недоліки. Так, наприклад, формат MP4 (MPEG-4) може використовуватись на багатьох видах пристроїв, має досить високий рівень стискання, що призводить до менших розмірів відео-файлів, а також дозволяє

додавати метадані до відеофайлів [17]. Тоді як формат MPEG-1 має значно менший рівень стискання даних, однак в нього є така особливість, як можливість конкатенації двох відеофайлів як двох звичайних файлів [18]. Тож давайте і скористаємось особливостями цих форматів.

Алгоритм інтеграції реклами у відео за допомогою фреймворку FFmpeg наступний:

1. конвертація відеоролику та відеореклами із вхідного кодування в кодування MPEG-1;
2. “розрізання” відеоролику на дві частини;
3. “зшивання” трьох відео у форматі MPEG-1 в один єдиний файл;
4. конвертація результуючого відео з формату MPEG-1 в MP4;
5. видалення всіх проміжних файлів.

Даний алгоритм можна подати у вигляді наступної послідовності виконання команд FFmpeg у командному рядку:

$ ffmpeg -i video.mp4 -c:v copy -c:a copy video.mpg
$ ffmpeg -i video.mpg -ss 00:00:00 -t T -c:v copy -c:a copy video-part-1.mpg
$ ffmpeg -i video.mpg -ss T -c:v copy -c:a copy video-part-2.mpg
$ ffmpeg -i ad.mp4 -c:v copy -c:a copy ad.mpeg
$ ffmpeg -i concat:”video-part-1.mpg|ad.mpg|video-part-2.mpg” result.mpg
$ ffmpeg -i result.mpg -c:v copy -c:a copy result.mp4

Більше того, існує бібліотека JavaScript, що виконує функції FFmpeg, що значно покращує якість коду.

2.4. Авторизація та аутентифікація користувачів

Надамо наступні два визначення.

Аутентифікація – процес визначення ідентичності користувача. Це механізм асоціювання вхідного запиту із множиною ідентифікуючих облікових даних [19].

Авторизація – механізм захисту, що визначає рівні доступу або привілеї користувачів відносно системних ресурсів, що включають в себе файли, сервіси, дані та функції програми [20].

Для роботи веб-застосунку необхідно реалізувати механізм авторизації та аутентифікації користувачів. В нашій роботі існує два види користувачів: рекламодавець та автор відео.

Наш сервіс буде працювати на основі середовища виконання NodeJS. Це потужне open-source, крос-платформенне середовище виконання коду JavaScript поза браузером. Open-source спільнота розробила безліч фреймворків для швидкої та ефективної роботи в цьому середовищі. Один з найпопулярніших фреймворків є Express.js, що дозволяє досить просто створювати Web-сервіси [21].

Для реалізації авторизації та аутентифікації користувачів ми будемо використовувати JWT токени [22]. Токен JWT складається з трьох частин: заголовка (header), корисного навантаження (payload) та підпису або даних шифрування.

Із можливих баз даних, що можуть бути використані для зберігання даних користувачів, враховуючи формат даних, оптимальним вибором буде база даних типу ключ-значення або ж документоорієнтована база даних. Враховуючи те, що розробка сервісу відбувається із використанням мови програмування JavaScript, пропонується використовувати найпопулярнішу документоорієнтовану базу даних, що дуже легко інтегрується з NodeJS додатками, – MongoDB [23].

Отже, суть операції реєстрації користувача полягає в виконанні запиту на HTTP-сервер із заданими обліковими даними: емейл, пароль, тип користувача (автор відео або рекламодавець), ім’я, прізвище і так далі. Ці облікові дані в процесі виконання запиту зберігаються в обрану раніше базу даних. Слід зауважити, що зберігання паролю в сирому вигляді є небезпечним з точки зору безпеки даних користувачів, оскільки банально доступ до бази даних може бути втрачений із-за зловмисників, які згодом зможуть отримати доступ до такої чутливої інформації як пароль. Тому для унеможливлення кражі паролю

користувача зазвичай, замість збереження паролю у сирому вигляді, зберігається його хеш.

Для виконання авторизації та аутентифікації користувача необхідно виконати процедуру логіну – виконати запит на HTTP-сервер з необхідними обліковими даними: емейл та пароль. В разі успішної аутентифікації користувача сервер повертає код успішного виконання запиту та http-only cookie з необхідною обліковою інформацією для подальшої авторизації.

Нижче наводиться приклад коду спеціальної проміжної функції, що використовується у нашому сервісі для подальшої авторизації.

```
export const auth = (config) => (req, res, next) => {
    const token = req.body.token || req.cookies.authcookie;

    if (!token) {
        return res.status(403).send("A token is required for authentication");
    }
    try {
        const decoded = jwt.default.verify(token, config.tokenKey);
        req.user = decoded;
    } catch (err) {
        return res.status(401).send("Invalid Token");
    }
  return next();
};
```

Як ми бачимо, в процесі виконання цієї функції відбувається розкодування JWT токену і його подальша перевірка на валідність. Згодом, у випадку успішної валідації токену, запит насичується додатковою необхідною інформацією про користувача, що згодом використовується для авторизації.

2.5. Виокремлення зображення з відео

В інтерфейсі для користувачів передбачено відображення відповідного зображення до відео (thumbnail), коли відео ще не було увімкнено для перегляду.

Для цього необхідно розробити функціонал, який буде конвертувати відео у зображення.

Для даної задачі можна використати сервіс AWS Elastic Transcoder. Цей сервіс надає можливість змінювати кодування та формат вхідного відео, а також створює необхідні нам зображення. Результат перекодування відео зберігається у відповідному бакеті AWS S3, тому питання зберігання вихідних зображень легко вирішується.

Процес виділення зображень відбуватиметься асинхронно, відразу після завантаження відео на платформу. Посилання із підписом на зображення буде віддаватись разом зі всією іншою інформацією про відео. У випадку, якщо процес виконання створення зображення ще не був завершений, буде повертатись стандартне зображення, що було окремо та завчасно завантажено на хмару.

2.5. Побудова RESTful сервісу

REST API – це такий API, що відповідає відповідає критеріям архітектурному стилю REST та дозволяє взаємодіяти з RESTful веб-сервісами [24].

Не варто плутати, що REST – це протокол або стандарт, бо насправді це лише множина архітектурних обмежень. Не існує якогось єдиного способу, яким такий API має бути реалізований розробником.

REST розшифровується як Representational State Transfer, тобто, з англійської, передача репрезентативного стану. По суті це значить, що, клієнт та сервіс оперують один з одним поняттями стану певної сутності або ж ресурсу. Коли клієнт робить запит на сервіс за допомогою RESTful API, він передає стан цього ресурсу. Стан ресурсу може бути описаний за допомогою різних форматів через HTTP: JSON, XTL, текст, HTML. JSON є загальноприйнятим форматом, оскільки він є агностичним до мови програмування та легко читається як людиною, так і машиною.

Для того, щоб API був RESTful, він має задовольняти наступним критеріям [24]:

1. архітектура клієнт-сервер, що реалізована за допомогою клієнтів, серверів, ресурсів, де запити виконуються за допомогою HTTP;
2. комунікація між клієнтом і сервером не має залежати від певного стану;
3. можливість кешування даних, що пришвидшує комунікацію між клієнтом і сервером;
4. наскрізний єдиний інтерфейс між компонентами, дані передаються у єдиній стандартній формі.

Для того, щоб виконувати операції над ресурсами в RESTful API, використовується протокол HTTP. Шлях до ресурсу задається за допомогою URI, а за допомогою HTTP методів (GET, PUT, POST) задається операція, що має бути виконана з ресурсом.

При розробці RESTful API притримуються правила, що в шляху до ресурсу мають бути тільки іменники, а за допомогою HTTP методів задається дієслово, яке визначає яка дія має бути виконана з ресурсом. Однак задля уточнення специфічної дії над ресурсом дозволяється в останній частині URI задавати назву конкретної операції, що буде задіяна над ресурсом.

Давайте визначимо які ресурси наявні в нашому сервісі: відео, користувачі, контракти. Відео може бути двох типів: відеоконтент та відеореклама. В той самий час користувачі також поділяються на два типи: рекламодавці та автори відео. Нижче наводиться визначення ресурсів сервісу із описом їх полів.

Таблиця 1.1 – Опис ресурсу відео

Назва поля	Опис
id	унікальний ідентифікатор відео, UUID рядок
userId	унікальний ідентифікатор користувача, що є власником відео, UUID рядок
title	назва відео

description	опис відео
downloadUrl	підписане посилання на відео, за допомогою якого його можна завантажити напряму з AWS S3
uploadUrl	посилання з підписом, за допомогою якого можна завантажити відео напряму на AWS S3
editedVideoId	посилання на унікальний ідентифікатор відео, що є результатом інтеграції реклами у відео, UUID рядок
isEdited	булеве значення, що показує чи була реклама інтегрована у відео
timestampsForAd	список часових кодів, які пропонуються для вставки реклами
thumbnail	посилання з підписом на картинку попереднього перегляду відео, яке можна завантажити напряму з AWS S3
type	рядок, що може приймати два значення: advertisement, content

Таблиця 1.2 – Опис ресурсу користувач

Назва поля	Опис
id	унікальний ідентифікатор користувача, UUID рядок
email	email користувача
firstName	ім’я користувача
lastName	прізвище користувача
type	рядок, що може приймати два значення: advertiser, contentMaker

Таблиця 1.3 – Опис ресурсу контракт

Назва поля	Опис
id	унікальний ідентифікатор контракту, UUID рядок
videoId	ідентифікатор відео, у яке пропонується інтегрувати рекламу
adId	ідентифікатор відео-реклами, яке пропонується інтегрувати

advertiserId	ідентифікатор користувача-рекламодавця
payment	розмір оплати за рекламну інтеграцію
status	статус виконання контракту, що має такі можливі значення: offeredByAdvertiser, acceptedByCreator, integratingAds, integratedAdsWaitingForApproval, approvedByAdvertiser, approvedByCreator, dealSigned, dealCanceled, errored

Нижче наводяться таблиці опису RESTful API над сутностями, що ми описали.

Таблиця 2.1 – Ендпоінти для сутностей користувачів

HTTP метод	Шлях	Опис
POST	/users/register	виконати реєстрацію нового користувача
POST	/users/login	виконати автентифікацію користувача
GET	/users/:id/videos	отримати список всіх відео, що належать користувачу з ідентифікатором id

Таблиця 2.2 – Ендпоінти для сутностей відео

HTTP метод	Шлях	Опис
GET	/videos	отримати список всіх відео поточного користувача
POST	/videos	створити нове відео
POST	/videos/:id/confirmVideoUpload	підтвердити завантаження відео на хмару
…		

Слід звернути увагу на процес створення та завантаження нового відео. Враховуючи те, що раніше було прийнято рішення завантажувати відео напряму з клієнта на хмару замість того, щоб спочатку завантажувати його на сервер, а потім

з сервера завантажувати його на хмару, що призводить до виконання зайвої роботи, процес завантаження відео з клієнта був розбитий на 3 етапи. Спочатку клієнт створює новий ресурс відео на стороні сервера, де з відповіді одразу можна отримати *uploadUrl*, що дозволяє завантажити відео на хмару, виконавши HTTP операцію *PUT uploadUrl.* Після успішного завантаження відео на хмару, клієнту необхідно повідомити про це сервер.

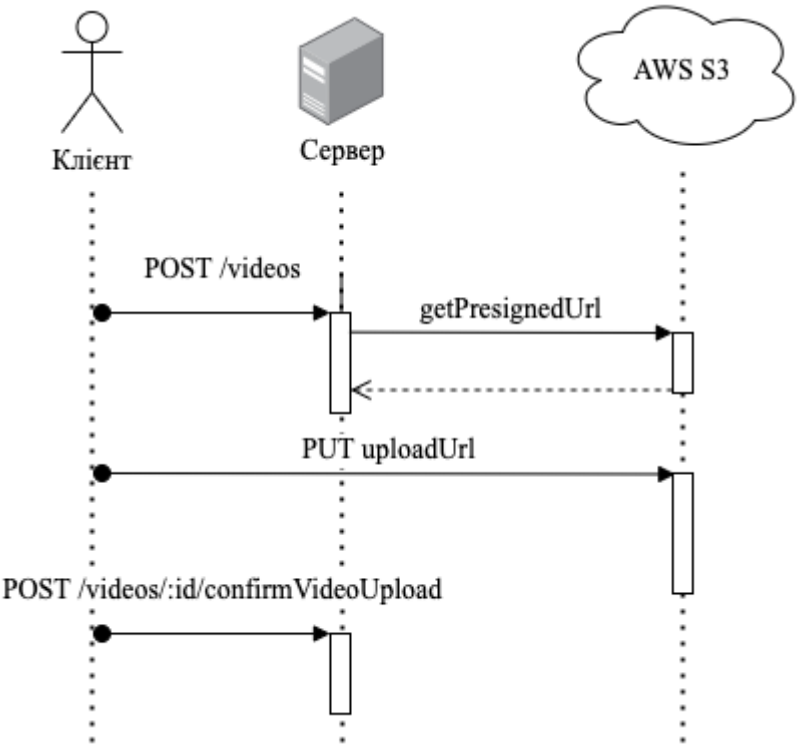

Рисунок 5 – Процес завантаження відео з використанням розробленого API

Для збереження метаданих відео, інформації про користувачів, контрактів було обрано документно-орієнтовану базу даних MongoDB із хмарним хостингом. Оскільки структура даних, з якими ми працюємо є досить простими, і немає потреби в складних запитах, що потребують об’єднань (join), документно-орієнтована є оптимальним рішенням для нашої задачі. Більше того, для подальшого пошуку відео за описом, дана база даних пропонує можливість використання спеціальних індексів для пошуку за текстом.

2.6. Діаграма компонентів

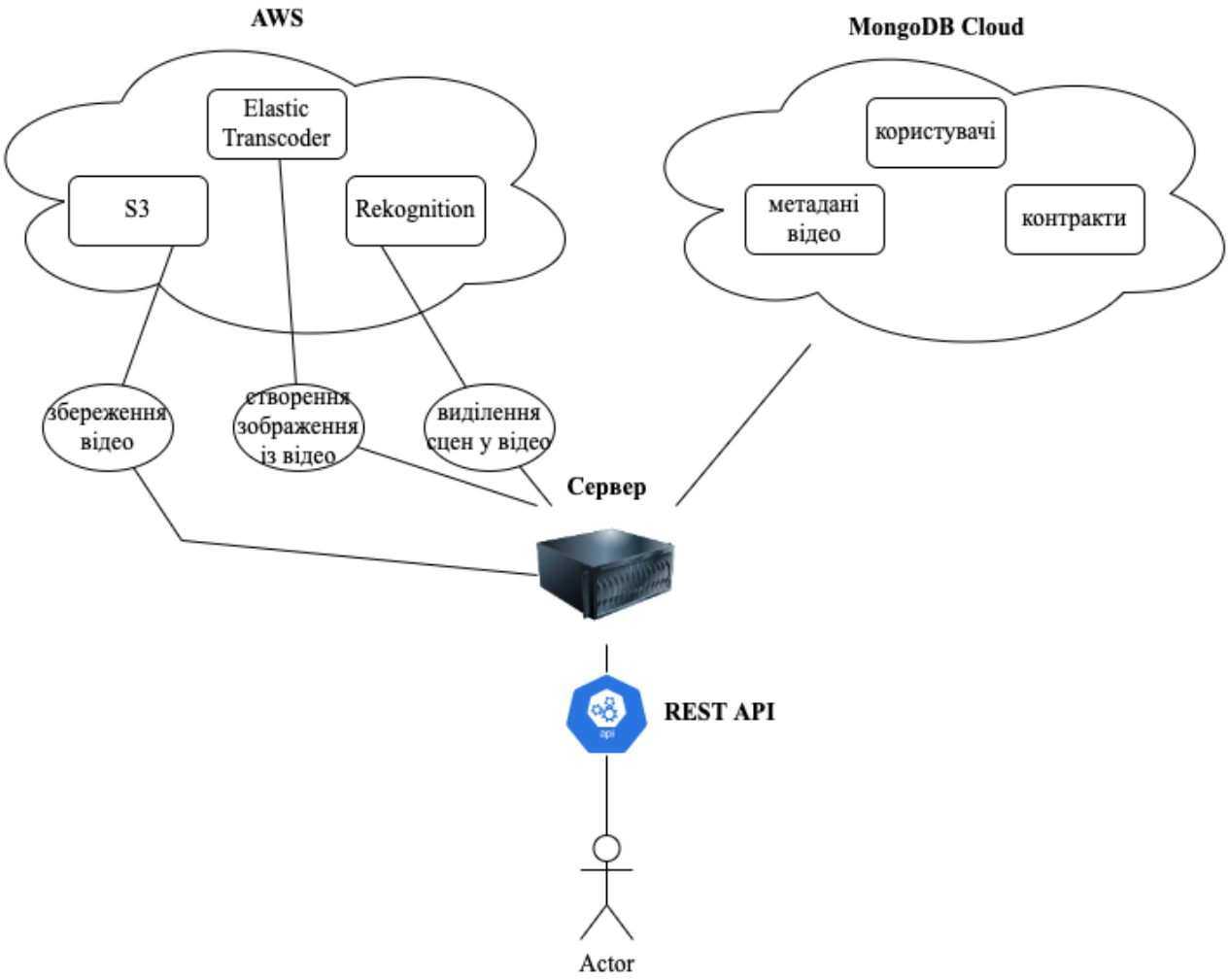

Рисунок 6 – Діаграма компонентів сервісу

РОЗДІЛ 3. ОГЛЯД ІНТЕРФЕЙСУ КОРИСТУВАЧА

3.1. Використані технології

Створення користувацького інтерфейсу може відбуватись декількома способами.

Перший спосіб – створення статичних сторінок на стороні сервера, які він згодом віддає на запит користувача. Для того, щоб відтворювати сторінки відповідно в залежності від отриманих даних, існують шаблонізатори – інструмент, що дозволяє задати певний шаблон HTML-сторінки, яка наповнюється даними, які їй надає сервер.

Другий спосіб – створення інтерфейсу за допомогою таких фреймворків як React та Vue. За допомогою них створюється окремий додаток, що живе незалежно від сервера. Даний підхід є більш сучасним, однак потребує більшої роботи.

Отже, для поставленої задачі у роботі, тобто для створення прототипу додатку, було обрано перший варіант – використання шаблонізатора, а саме движок шаблонів Pug [27].

3.2. Сторінка реєстрації

Першою сторінкою, куди потрапляє користувач, є сторінка реєстрації, на якій він має ввести свої персональні дані: ім’я, прізвище, поштова скринька, пароль. Також користувач має обрати який тип акаунту він хоче створити – акаунт автора відео (Content Maker) або рекламодавця (Advertiser).

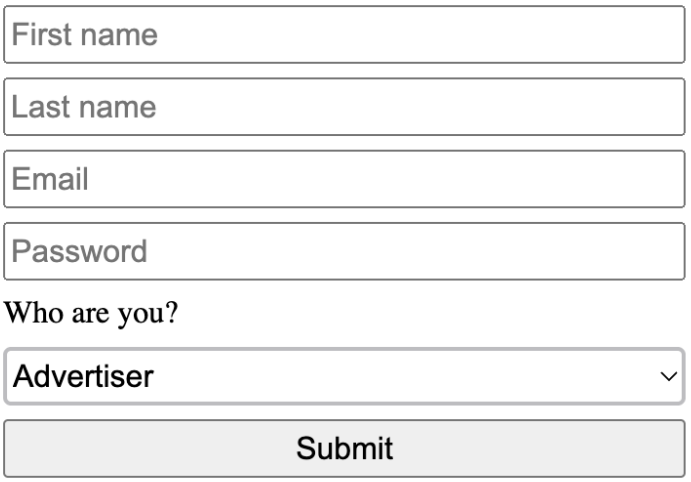

Рисунок 7 – Форма реєстрації користувача

3.3. Сторінка аутентифікації

Сторінка аутентифікації має форму подібну до форми реєстрації.

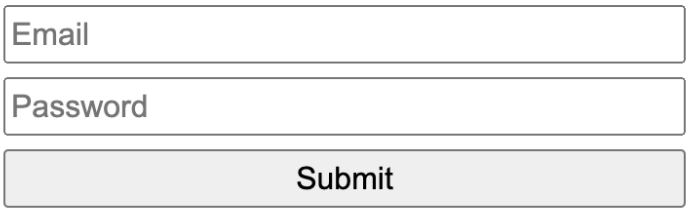

Рисунок 8 – Форма аутентифікації користувача

3.4. Головна сторінка та сторінка перегляду відео

На головній сторінці користувач може побачити форму для завантаження власного відео або реклами в залежності від типу користувача, а також список завантажених ним раніше відео. Користувач відкрити обране власне відео для перегляду.

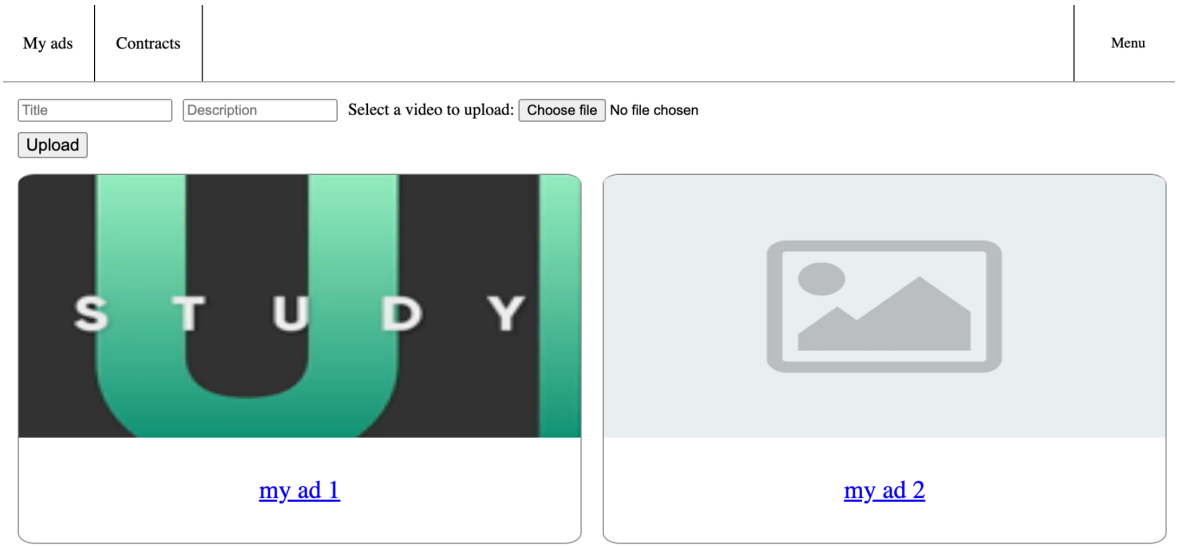

Рисунок 9 – Головна сторінка користувача

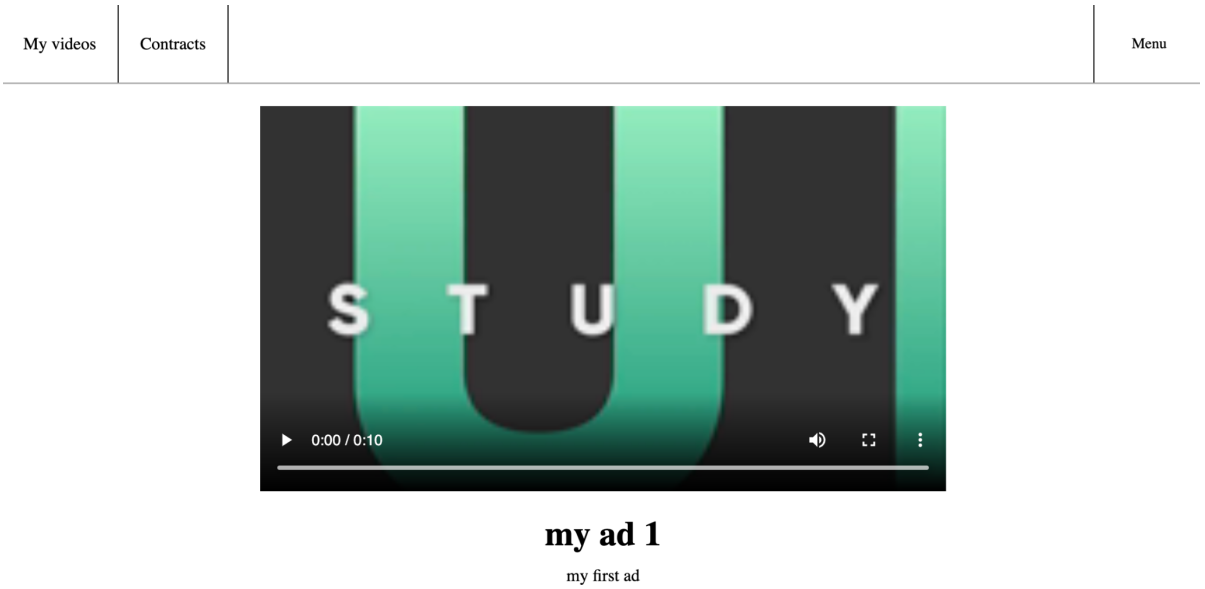

Рисунок 10 – Сторінка перегляду відео

3.5. Сторінка контрактів та створення нових контрактів

На сторінці наявних контрактів обидва типи користувачів можуть побачити список власних контрактів, що були створені власноруч або прийняті від іншої сторони. Можна побачити статус контракту та результат роботи, якщо контракт

був виконаний. Також є окрема сторінка для створення нових контрактів із зазначенням винагороди за виконання контракту.

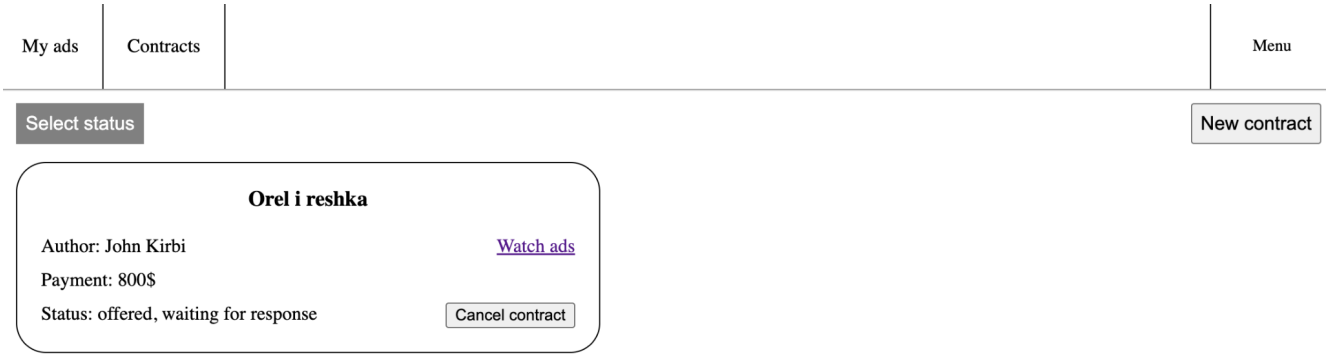

Рисунок 11 – Сторінка перегляду наявних контрактів

РОЗДІЛ 4. РОЗГОРТАННЯ ДОДАТКУ

Невід’ємною частиною розробки сервісу є його розгортання – сукупність дій, що має бути виконана для того, щоб додаток став готовим для користування для кінцевого користувача. У випадку веб-сервісу необхідно, щоб сервіс був доступний для користувача через мережу Інтернет. У даному розділі ми розглянемо розгортання розробленого нами RESTful сервісу за допомогою таких сервісів AWS: Elastic Compute Cloud, Elastic Container Service, Lambda [28, 29, 30].

4.1. Віртуальний сервер

Найпростішим способом розгортання додатку є його запуск із усіма необхідними конфігураціями такими як опис доступу до сервісу з мережі, приватні ключі необхідні для роботи сервісу, тощо на виділеному сервері в єдиному екземплярі.

Даний спосіб розгортання можна реалізувати за допомогою хмарного сервісу Elastic Compute Cloud. Він надає віртуальні сервери, які можна використовувати для будь-яких потреб, що не порушують правил користування даним сервісом. Існує дуже велика варіативність вибору серверу під потреби будь-якого випадку. В тому числі для розгортання веб-сервісів. Слід зауважити, що даний спосіб можливо реалізувати далеко не тільки за допомогою сервісів Amazon. Є такі аналоги даного сервісу: Google Cloud Platform, Microsoft Azure, Digital Ocean, тощо.

Головним недоліком даного способу розгортання є те, що в якийсь час сервер може досягти свого ліміту по ресурсам і його буде вкрай важко розширити – додати нових потужностей або таких ресурсів як оперативна пам’ять або більш потужний процесор. Тобто сервіс складно масштабувати “вертикально”.

4.2. Контейнеризація

Більш оптимальним та найбільш популярним способом розгортання веб-додатків на сьогодні є розгортання веб-сервісів за допомогою контейнеризації. Контейнеризація полягає в пакуванні виконуваного коду разом з бібліотеками та залежностями операційної системи, необхідних для запуску коду, щоб створити легкий виконуваний файл, названий контейнером, що може бути запущеним на будь-якій інфраструктурі.

Концепт контейнеризації має свій початок ще у давні часи, однак особливої популярності він почав набирати з 2013 року, моменту появи стандарту контейнерів – Docker Engine.

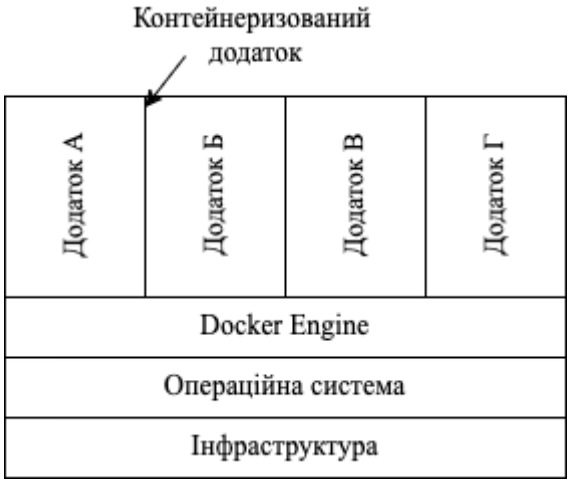

Рисунок 12 – Схематичне зображення роботи контейнерів

Elastic Container Service – це сервіс, що надає послуги з оркестрації, тобто автоматичної конфігурації, менеджменту, балансування навантаження та координації, контейнерів Docker.

Перевагами використання даного методу є:

1. легкість масштабування сервісу “в ширину” – все, що для цього потрібно, це збільшення кількості екземплярів додатку;
2. спрощення розробки та тестування програмного забезпечення, бо зникає необхідність налаштовувати середовище для тестування;
3. ефективніше та ізольоване використання ресурсів.

4.3. Serverless

Serverless – модель виконання коду у хмарі, що не потребує постійної роботи серверу. Його суть полягає в тому, що хмарний сервіс виділяє ресурси на вимогу кожного виклику функції. Таким чином користувач не має потреби обслуговувати сервер, і зникає проблема марної витрати ресурсів у період “простою” серверу – користувач платить лише за реально використані ресурси, що були необхідні для виконання певної функції.

Даний метод найкраще підходить для таких типів задач:

1. фонові задачі, що потребують великої кількості витраченого часу, як наприклад обробка медіафайлів;
2. клієнто-орієнтовані додатки, де більшість логіки може бути перенесена на сторону клієнта;
3. розробка додатків, що мають швидко масштабуватись та змінювати виконувані функції.

Одним із таких хмарних сервісів є AWS Lambda. Цей сервіс дозволяє запускати обчислення на основі serverless моделі. Даний сервіс дозволяє додавати функції, що будуть виконуватись в залежності від певної події. Це може бути створення нового об’єкту в бакеті S3, або ж оновлення значення для певного ключа в сховищі типу ключ-значення DynamoDB.

У розробленому нами додатку є функціонал, що міг би бути реалізований за допомогою цієї технології: створення зображення для попереднього перегляду відео та відправка запиту аналізу відео для виявлення у ньому сегментів сцен. У даному випадку функція викликалась би кожного разу, коли у відповідному бакеті створювався новий об’єкт-відеофайл.

Сервіс AWS Lambda можна легко використовувати для розробки додатків, де основна частина логіки знаходиться на клієнтській стороні. Сервіс має багатий на функціонал та інтуїтивний інтерфейс, що звільняє від необхідності витрачати багато часу на створення та конфігурування функцій.

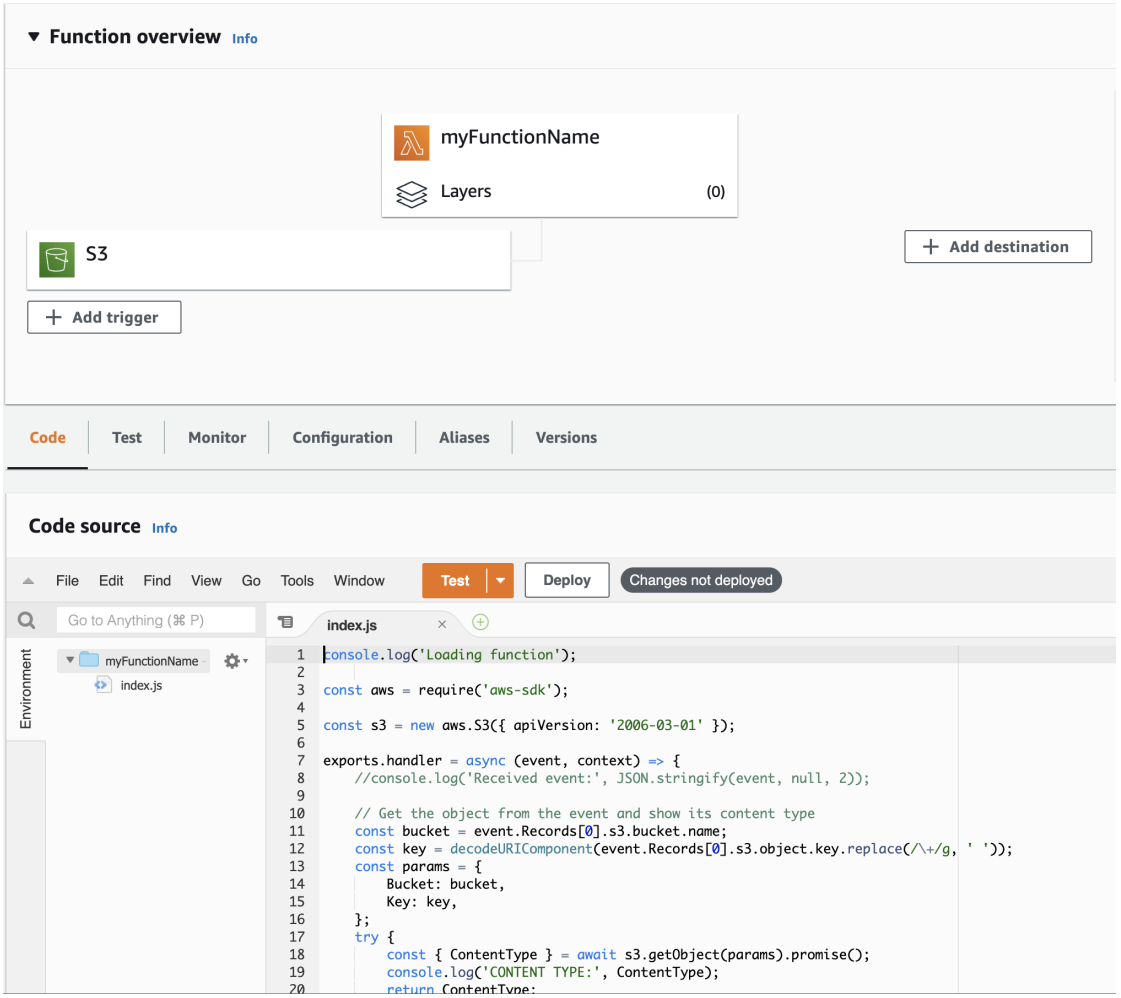

Рисунок 13 – Інтерфейс сервісу AWS Lambda

ВИСНОВКИ

У даній роботі було поставлено мету розробити прототип веб-платформи, що дозволяє рекламодавцям та авторам відеоконтенту укладати рекламні договори та створювати рекламні інтеграції у відео.

В ході роботи було проведено аналіз поточного стану ринку цифрової реклами, в результаті якого було отримано висновки, що даний ринок є швидко зростаючим.

У процесі виконання даної роботи було досліджено та реалізований наступний функціонал сервісу:

1. авторизація та аутентифікація користувачів;
2. завантаження та збереження відео;
3. укладення рекламних контрактів між рекламодавцем на виконавцем;
4. рекомендації часових кодів у відео для рекламної інтеграції;
5. автоматична інтеграція реклами у відео.

Також була досліджена тема розгортання розробленого веб-сервісу 3 методами: на віртуальному сервері, за допомогою контейнеризації та за допомогою технології serverless.

Було досліджено наступні хмарні технології AWS:

1. Simple Storage Service;
2. Elastic Transcoder;
3. Rekognition Video;
4. Elastic Compute Cloud;
5. Elastic Container Service;
6. Lambda.

Також було досліджено та використано документно-орієнтовану базу даних MongoDB Atlas.

Створений веб-додаток дозволить створити єдину платформу для укладання рекламних договорів між рекламодавцями та авторами відео, що значно зменшить кількість роботи для обох сторін.

Більше того, було закладено ідеї для подальшого розвитку проекту. Можливо реалізувати функції створення зображень попереднього перегляду та створення запиту на аналіз відео за допомогою технології serverless. Також можливе використання хмарних технологій для реалізації функціоналу автоматичної інтеграції реклами у відео. Більше того, є необхідність в інтеграції з платіжним провайдером для повноцінної реалізації оплати договорів, що є окремим складним завданням як з продуктової точки зору, так і з технічної.

Використовуючи розроблений прототип можна продовжувати розвиток веб-додатку із додаванням нового функціоналу та покращенням технічних показників сервісу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Global internet advertising revenue from 2020 to 2025 [Електроний ресурс] – Режим доступу до ресурсу: https://www.statista.com/statistics/237800/global-internet-advertising-revenue/
2. Worldwide Digital Ad Spending Year-End Update [Електроний ресурс] – Режим доступу до ресурсу: https://www.emarketer.com/content/worldwide-digital-ad-spending-year-end-update
3. Global internet advertising revenue from 2020 to 2025 [Електроний ресурс] – Режим доступу до ресурсу: https://blog.hubspot.com/marketing/state-of-video-marketing-new-data
4. What Video Marketers Should Know in 2022, According to Wyzowl Research [Електроний ресурс] – Режим доступу до ресурсу: https://www.statista.com/statistics/351862/adblocking-usage/#:~:text=Adblocking%3A%20penetration%20rate%202020%2C%20by%20country&text=The%20average%20global%20adblocking%20rate,of%2056.8%20and%2050.7%20percent.
5. Base64 [Електроний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Glossary/Base64
6. HDFS Architecture Guide [Електроний ресурс] – Режим доступу до ресурсу: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html
7. Amazon S3 [Електроний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/s3/
8. Top 5 Reasons for Choosing S3 over HDFS [Електроний ресурс] – Режим доступу до ресурсу: https://databricks.com/blog/2017/05/31/top-5-reasons-for-choosing-s3-over-hdfs.html
9. PySceneDetect [Електроний ресурс] – Режим доступу до ресурсу: http://scenedetect.com/en/latest/

10. Scene Detection Algorithms [Електроний ресурс] – Режим доступу до ресурсу: http://scenedetect.com/en/latest/reference/detection-methods/
11. RGB color model [Електроний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/RGB_color_model
12. HSL and HSV [Електроний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/HSL_and_HSV
13. Amazon Rekognition Video [Електроний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/rekognition/video-features/
14. Detecting video segments in stored video [Електроний ресурс] – Режим доступу до ресурсу: https://docs.aws.amazon.com/rekognition/latest/dg/segments.html
15. Amazon Elastic Transcoder [Електроний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/elastictranscoder/
16. About FFmpeg [Електроний ресурс] – Режим доступу до ресурсу: https://ffmpeg.org/about.html
17. Learn more about the MP4 video format [Електроний ресурс] – Режим доступу до ресурсу: https://www.adobe.com/creativecloud/video/hub/features/learn-about-mp4-format-videos
18. How can I join video files[Електроний ресурс] – Режим доступу до ресурсу: https://ffmpeg.org/faq.html#How-can-I-join-video-files_003f
19. What is 'Authentication' [Електроний ресурс] – Режим доступу до ресурсу: https://economictimes.indiatimes.com/definition/authentication
20. What is 'Authorization' [Електроний ресурс] – Режим доступу до ресурсу: https://economictimes.indiatimes.com/definition/authorization
21. Express.js [Електроний ресурс] – Режим доступу до ресурсу: https://expressjs.com/
22. JSON Web Token (JWT) [Електроний ресурс] – Режим доступу до ресурсу: https://datatracker.ietf.org/doc/html/rfc7519

23. MongoDB Atlas [Електроний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/atlas/database
24. What is REST API [Електроний ресурс] – Режим доступу до ресурсу: https://www.redhat.com/en/topics/api/what-is-a-rest-api
25. Segmentation and Classification with HSV [Електроний ресурс] – Режим доступу до ресурсу: https://medium.com/neurosapiens/segmentation-and-classification-with-hsv-8f2406c62b39
26. Introducing TikTok Branded Mission: Inspiring Brand and Creator Collaborations [Електроний ресурс] – Режим доступу до ресурсу: https://newsroom.tiktok.com/en-us/introducing-tiktok-branded-mission-inspiring-brand-and-creator-collaborations
27. Pug.js [Електроний ресурс] – Режим доступу до ресурсу: https://pugjs.org/api/getting-started.html
28. Amazon EC2 [Електроний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/ec2/
29. Amazon ECS [Електроний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/ecs/
30. Amazon Lambda[Електроний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/lambda/