Приватний заклад вищої освіти
Одеський технологічний університет «ШАГ»
Кафедра інформаційних технологій та фундаментальної підготовки

Випускна кваліфікаційна робота бакалавра

«СИСТЕМА КЕРУВАННЯ ПОТОКОВИМ КОНТЕНТОМ (за прикладом сервісу Youtube)»

на здобуття бакалаврського рівня вищої освіти
зі спеціальності «122 Комп'ютерні науки»

Виконавці проекту:

№	П.І.Б.	Група	Роль
1	Головань В. В.	КН-П-181	Розробка (frontend)
2	Чайковский А. В.	КН-Д-182	Розробка UI/UX - дизайну
3	Байдуков М. М.	КН-П-181	Розробка (backend)

Автор звіту

_____ Головань Владислава Володимирівна

Одеса – 2022

АНОТАЦІЯ

УДК 004.9
ГГ-6161

Головань В. В. Система керування потоковим контентом: Автореферат випускної кваліфікаційної роботи на здобуття бакалаврського рівня вищої освіти зі спеціальності «122 Комп'ютерні науки». - Одеса: ОТУШ, 2022 - 13 с.

Робота була проведена з метою дослідити та розробити інформаційну систему, головним призначенням якої було б керування потоковим контентом, а саме: створення, актуалізація, модифікація, відтворення та його знищення. Додатковими функціями є: групування, різні види фільтрації та демонстрації контенту, можливість його оцінювання, коментування та розповсюдження, ведення статистики, реєстрація, авторизація та автентифікація.

Актуальність роботи визначається необхідністю популяризувати відеоконтент серед великої кількості людей простим шляхом за невеликий проміжок часу, просування бізнесу/брендів або монетизація розважливого контенту задля впровадження реклами і т.п. Особливу увагу звертає на себе той факт, що в основі проекту лежить алгоритм, який, аналізує дії та зберігає відповідні уподобання користувача на платформі. Це дозволяє в майбутньому формувати контент, підходящий саме для цього користувача і так для кожного. Додатково, актуальність диктується активним розвитком відео-медійного простору, в якому контент за невеликий відрізок часу освітлює максимум потрібної інформації на велику аудиторію. А через те, що платформа адаптується й під телефон, то це робить її ще більш доступною та зручною.

Об'єктом дослідження у рамках даної роботи виступає потоковий контент та сфера його розповсюдження. Предметом же є керування, поширення та фільтрація цього контенту. У якості задач дослідження було необхідно:

- провести аналіз популярних відео платформ, виявити особливості, характерні риси та напрями удосконалення.
- визначити програмні засоби (інтерфейси API), які потрібні для функціонування проекту, встановити ступінь вживаності для поставлених завдань.
- розробити програмне забезпечення на основі двох вищевказаних пунктів, адаптувати під різні пристрої та протестувати дієздатність на кожному з них.
- впровадити у систему нові корисні функції, які будуть робити платформу ще більш зручнішою для користувача та легкою у користуванні.

Методи дослідження, які були застосовані задля досягнення поставлених задач: системний аналіз, моделювання та когнітивна методологія.

При складанні звіту використано посилання на друковані та електронні джерела інформації.

ЗМІСТ

ВСТУП

Стрімкий розвиток медіа простору диктує нам не відставати та “йти в ногу” з новими тенденціями, а отже розробка відео платформи є дуже актуальною. З кожним днем такого виду контент дивляться все більше, і більше людей, а активні бренди за це отримують прибуток.

Також важливо помітити, що відео можна переглянути на будь-якому мобільному пристрої. Тобто доступність швидкої легкозасвоюваної інформації дійсно приваблює величезну кількість людей. Адже темп життя дуже швидкий, тому все менше хотілось би себе чимось уповільнювати, а інформаційні відео платформи не дають цього зробити своєю зручністю та лаконічністю інформації.

Таким чином, успішний бізнес, який зосереджений на створенні відеоконтенту, охоплює максимальну кількість людей. Тобто, розробкою та розвитком таких платформ зацікавлені діячі, бо з їхнім використанням можна легко просувати свій бізнес, збільшувати свою аудиторію, популяризувати бренд чи впроваджувати рекламу.

Мобільні пристрої, - смартфони, планшети, ноутбуки, - стали невід’ємною частиною сучасної людини. А отже, з’явилися нові професії зв’язані з цими гаджетами. Завдяки відео платформам, людина набуває можливостей: покращення кваліфікації або й зовсім опанування нового, проявлення творчості у створенні контенту, набуття та зближення зі своєю аудиторією, особливо, якщо це бізнес-діячі. Таким чином, можна зробити висновок, що подібні технології широко проникли майже в всі галузі діяльності та продовжують це робити.

У той же час, широкий вжиток інформаційних систем потребує розробки з покращеними умовами зручності користування та з індивідуальними алгоритмами для кожного. Адже створити нове удосконалене набагато ефективніше, ніж на базі старого.

Предметом даної роботи було обрано керування частиною життєвого цикла потокового контенту, а саме: зберігання, відтворення, поширення та видалення.

Також функції переважної більшості подібних систем мають передбачати можливість збереження історії перегляду, пошуку відео, а також різні способи фільтрації та демонстрації. У перспективі — впровадження реклами та монетизацію контенту. Відповідно, це передбачає засоби авторизації та автентифікації, а також убезпечення персональних даних від несанкціонованого доступу інших користувачів.

Детальне технічне завдання до проекту наводиться далі. У рамках даного звіту розглядається клієнтська частина, що являє собою сайт, адаптований під різні пристрої.

Даний звіт об’єднує у собі три розділи: загальний опис системи, деталі створення клієнтської частини, результати її працеспроможності та тестування.

В кінці звіту, підведені висновки, що підкреслюють результати роботи даної частини проекту, а загальний висновок є результатом роботи усіх учасників проекту.

ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОЕКТУ

Загальний опис.

Програмне забезпечення (ПЗ) призначене для створення розподіленої системи управління контентом, який має потоковий зміст. Головна особливість системи полягає у тому, що наявний контент має бути забезпечений засобами відтворення (перегляду або прослуховування) без необхідності завантаження. Функції системи мають забезпечити частковий життєвий цикл управління контентом: публікація, актуалізація, модифікація та знищення.

Призначення:

ПЗ орієнтується на створення узагальненої концепції контейнера для потокових даних. Головний акцент робиться на засобах групування, пошуку та розподілу доступу до контенту. Функції відтворення контенту не є безпосереднім предметом завдання і можуть бути запозичені у третіх сторін. ПЗ створюється за прикладом сервісу відеохостингу, що надає послуги розміщення відеоматеріалів, у той же час має складатись за принципом універсальності, мати змогу змінити цільове призначення.

Вимоги до функціоналу. Обов'язкова частина

Головна сторінка

Стартовий інтерфейс ПЗ (головна сторінка або активність) має за мету популяризацію системи. На ньому має бути забезпечено відображення популярних показників, як-то загальна кількість користувачів та інформаційних одиниць, вибірка з найбільш популярного контенту та нових дописів, зовнішня реклама. Посилання на інші сторінки (засоби) системи, а також на сторінки партнерів.

Реєстрація користувачів:

Користувач ПЗ має змогу зареєструватись як за допомогою авторизаційних даних (логіну/паролю), так і за допомогою облікових записів популярних соціальних мереж (Google, Facebook, тощо).

При реєстрації обов'язково зазначається засіб оперативного контактування (ЗОК) - телефон та електронна пошта. На тестовому етапі, за умови ускладнення замовлення послуг телефонії чи СМС, можливе використання лише електронної пошти. Засіб оперативного контактування при реєстрації підлягає верифікації через відповідь на тестовий виклик чи надісланий лист, або введення коду, пересланого через повідомлення довільного типу. Рекомендується використовувати ЗОК у якості авторизаційного логіну користувача.

За умови що користувач забув пароль, ЗОК має бути використаний для його відновлення. Схема відновлення паролю жорстко не регламентується.

Також передбачається наявність у системі користувачів з додатковими правами: модераторів та адміністраторів. Реєстрація кореневого адміністратора системи має бути реалізоване при інсталяції системи, наступним користувачам змінюють права наявні адміністратори.

Управління власним контентом:
Основний ресурс зареєстрованого користувача ПЗ. Повинен забезпечити усі необхідні засоби для

- Створення даних та метаданих (розміщення відео та його характеристик). Серед рекомендованих характеристик: опис, режим доступу (загальний/приватний), категорія, теги, вікові обмеження, дозвіл/заборона коментування.
- Встановлення, зміна та вилучення метаданих у власному контенті (редагування)
- Групування контенту (за принципом плейлистів)
- Перегляд поточної активності: історії переглядів, перелік відзначеного контенту (вибране), контактні особи (друзі), коментарі до контенту, тощо
- Пошук контенту інших користувачів за метаданими, групування та фільтрування результатів за різними ознаками, наявними у знайденому контенті. Для об'ємних результатів має бути забезпечена пагінація.
- Контактування з іншими користувачами, що розміщують контент, за допомогою ЗОК або через внутрішні засоби системи (чат або)

Також засобами особистого кабінету має бути передбачено можливість зміни персональних даних. У разі зміни ЗОК необхідна його актуалізація за тією ж схемою, що й при реєстрації.

Зворотній зв'язок щодо контенту
Користувач має змогу встановити власну оцінку (лайк або бал) для довільного контенту. Якщо дозволено коментування контенту, користувач має змогу залишити коментар. Дозволяється встановлювати обмеження на кількість коментарів від одного користувача. Засоби пошуку, фільтрування та групування мають забезпечити врахування рейтингу контенту, у т.ч. наявність чи кількість коментарів, наприклад, виводити результати у порядку за якого першими йдуть найбільш коментовані одиниці.

Вимоги до функціоналу. Опціональна частина

Модерація
Для забезпечення дотримання загальноприйнятих норм спілкування, а також з метою недопущення публікування контенту, який містить заборонені компоненти (ненормативні висловлювання, заклики до протиправних дії, приниження чи дискримінація, тощо), усі інформаційні одиниці, а також створені користувачами метадані (описи, теги), а також коментарі мають можливість бути заблокованими. Право блокування належить кореневому адміністратору системи, а також модераторам. При погодженні інформаційна одиниця має зберігати ідентифікатор користувача, який встановив блокування. Також має бути зворотна можливість розблокування заблокованого контенту.

Привілейовані функції
З метою покращення економічної ефективності проекту передбачається наявність додаткових функцій системи, що доступні лише за умови платної підписки (або внутрішніх досягнень) користувачів. До таких функцій можна

віднести відображення ЗОК інших користувачів (для прямого контактування), розширені засоби фільтрації та групування. Також можливе встановлення обмеження щодо кількості опублікованих одиниць контенту, коментарів, груп (плейлистів) тощо.

Архітектура

ПЗ має бути спроектоване за архітектурою PWA (Progressive Web Application) з чітко відокремленими розподіленими вузлами системи:

Mockup - візуальна модель інтерфейсу користувача із зазначенням зв'язків і переходів між різними частинами інтерфейсу.

Backend - серверне програмне забезпечення, що забезпечує збереження та оброблення поточних даних, взаємодію з базою даних. Надає програмний інтерфейс взаємодії (API) з іншими вузлами.

Web App - додаток, доступний з веб-простору (сайт), що забезпечує візуальний інтерфейс користувачів усіх категорій

Mobile App (опціонально) - додаток, що встановлюється на мобільних пристроях (смартфон, планшет). Дана частина виконується за наявності у команді проекту трьох і більше програмістів.

.

Усі модулі ПЗ (окрім візуальної моделі Mockup) повинні обов'язково включати до свого складу засоби

- тестування функціональності та модульні тести,
- розгортання на новому пристрої (інсталяції) із встановленням початкових таблиць у БД та кореневого користувача-адміністратора

Технології

Зберігання вихідних кодів ПЗ повинно здійснюватись за допомогою системи контроля версій (вибір конкретної системи узгоджується групою проекту). Структура файлів та каталогів повинна відбивати архітектуру ПЗ, на зразок:

- root
    - Backend
    - Web
    - Mobile
    - Mockup (вихідні файли або посилання на зовнішній проект)

Для кожного з модулів рекомендується (з можливістю часткової зміни) наступні технології створення:

Backend - PHP технології із виділеним або віртуальним хостингом,

Web - JS, HTML, CSS,або JS фреймворки, зокрема ReactJS, AngularJS

Mobile - Java, або Kotlin, або Flutter

Mockup - визначається ментором з дизайну

Команда проекту

Для виконання поставлених завдань передбачається робота команди розробників окремо за кожним з модулів проекту.

Мінімальна команда проекту - два учасники, у такому складі один учасник виконує модуль Backend та програмну частину Web, другий - модель Mockup та розміткову частину Web.
Рекомендована команда проекту - три учасники, у такій команді кожен з учасників відповідає за одну з частин: Mockup, Backend, Web.
Дозволяється включення до команди більшої кількості учасників, у такому разі модуль Mobile App набуває обов'язкової реалізації. Також модуль Mockup може бути поділений на окремі моделі для Web та Mobile частин.

Звітність

За результатами виконання завдань із створення ПЗ кожен з учасників команди проекту складає власний (індивідуальний) звіт, що складається з практичної частини та автореферату. Практична частина є прямим результатом роботи і подається у вигляді посилання на репозиторій (чи інший URL) або у вигляді архіву на довільному носієві даних. Вихідний код повинен оформлятися у відповідності до загальноприйнятих стандартів, на зразок стандартів GNU http://www.gnu.org/prep/standards/, у тому числі з додержанням вимог до коментування та документування коду.
Автореферат містить описову частину, у якій розкриваються основні концепти проекту, деталізуються завдання до виконавця, встановлюється актуальність роботи, її об'єкт та предмет, обґрунтовується вибір технологій та відзначається шлях виконання та ступінь досягнення поставлених задач, посилання на джерела інформації.

РОЗДІЛ 1
ЗАГАЛЬНА ХАРАКТЕРИСТИКА СИСТЕМИ

У відповідності до технічного завдання (ТЗ) система була спроектована за архітектурою «Progressive Web Application» (PWA). У складі системи було умовно відокремлено наступні архітектурні частини:

- «Mockup»,
- «Backend»,
- «Frontend».

Частина «Mockup» являє собою набір моделей та описів поведінки користувача. Модель подає інтерфейси взаємодії користувачів різного типу з інформаційною системою, траєкторії рухів та переходів між різними інтерфейсами. При створенні інтерфейсів застосовано принципи розумної навігації, включно із підходами UI/UX, висновки колористики та ергономіки для різних видів пристроїв: смартфони, планшети, лептопи, настільні комп'ютери. Макети цієї роботи можна переглянути за даним посиланням: https://www.figma.com/file/NWgcgIlTsZgB1ugP0pxNNm/Design-FunTab?node-id=0%3A1.

Частина «Backend» реалізована як централізований вузол збереження та оброблення даних, що створюються та модифікуються протягом життєвого циклу інформаційної системи. Дана частина включає у себе базу даних проекту та програмний інтерфейс (API) доступу до основних її функцій, зокрема, керування контентом, авторизації, узгодження профілів користувачів різного типу та/або різних пристроїв. Частина «Backend» виконана на базі програмного забезпечення PHP та СУБД «MySQL».

Клієнтська частина системи «Front-end», призначена для надання користувачеві візуального інтерфейсу через який він отримує доступ до функцій керування потоковим контентом розробленої інформаційної системи. Також до функцій даної частини належить збереження локальних даних для можливості використання системи при повторному вході без повторної авторизації. Вихідні коди та ресурси даної частини можна переглянути в репозиторії за даним посиланням: https://VladislavaVG@bitbucket.org/VladislavaVG/funtab.git. Подальший звіт присвячений детальному опису цієї частини. Частина «Front-end» реалізована на базі фреймворку React з використанням бібліотек Redux(для гнучкої взаємодії з даними в проекті та їх зберігання), а також Material Design(для реалізації такого ж гнучкого візуального інтерфейсу).

З метою покращення безпеки, звертаючи увагу на той факт, що дані, які передаються при взаємодії «Backend» і «Frontend» частин, легко можуть бути перехопленими зловмисниками, було прийняте рішення: впровадити шифрування криптоалгоритмом AES-128. Адже, він швидкий та ефективний, (що досить важливо для системи керування потоковим контентом), а також ймовірність отримати наслідки від повної атаки на сайт(якщо вона буде зроблена) — зовсім маленька, адже даний алгоритм має сильний розклад ключів.

РОЗДІЛ 2
РЕАЛІЗАЦІЯ КЛІЄНТСЬКОЇ ЧАСТИНИ

Клієнтська частина інформаційної системи реалізована з використанням наступних технологій:

- Фреймворк React v18.0
- Засоби розроблення: мова програмування JavaScript, середа розробки Virtual Studio Code
- Бібліотеки: Redux, React Redux, Redux Form, Axios, Material UI
- Додаткове ПЗ та npm-пакети: Google-API для авторизації через Google-акаунт та розпізнавання голосу для голосового ввода тексту

У складі клієнтської частини (адаптивного сайту) передбачено та реалізовано наступні функції:

- Авторизація користувача та реєстрація нових,
- Публікація, відтворення, модифікація та знищення потокового контенту,
- Можливість його групування, фільтрації та оцінювання.

Початок роботи користувача сайту починається з реєстрації як учасника відеоплатформи. Це можна зробити або через Google-акаунт [7], або напряму через форму реєстрації, яка пропонується сайтом. При реєстрації користувач обов'язково надає відомості про електронну пошту, після чого на неї надсилається верифікаційний код. Цей код потрібно ввести в форму [4] на сайті для перевірки пошти, і тільки тоді, коли код підтвердиться, користувач вважається зареєстрованим. У разі реєстрації через Google-акаунт, цей крок пропускається. Надалі, в перспективі розробки, дана пошта буде використовуватися задля розсилки повідомлень або пропозицій та користувач зможе відписатися від цієї функції. Попередньо до реєстрації введені дані проходять попередню валідацію за допомогою засобів регулярних виразів в полях вводу значень:

```
let emailPattern = `^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$`;
let namePattern = `^[0-9A-Za-z-_\.]{8,20}$`;
let passwordPattern =
`^[0-9A-Za-z-_\.`~!@#$%^&*()+=\?]{8,20}$`;
```

Авторизація користувача здійснюється за допомогою персонального секрету (логіну та паролю), введеного на етапі реєстрації чи через інші засоби авторизації [7]. Наприклад, через Facebook чи Google акаунтів, але тоді пароль не потрібен.

При реєстрації чи авторизації здійснюється комунікація з частиною «Backend» у результаті чого авторизація підтверджується або спростовується. Дані, що передаються між зазначеними частинами, шифруються за допомогою алгоритму AES-128 (режим CBC) та перетворюються у транспортне кодування Base64. Це убезпечує дані при передаванні. Для використання шифрування реалізоване окремим скриптом:

```
function SubmitsEncry(txtUserName, txtpassword)
{
        if (txtUserName.length == 0) {
            return false;
        }
        else if (txtpassword.length == 0) {
            return false;
        }
        else {
            let encrypted = [];
            const key =
CryptoJS.enc.Utf8.parse('8080808080808080');
            const iv =
CryptoJS.enc.Utf8.parse('8080808080808080');

            let encryptedlogin =
CryptoJS.AES.encrypt(CryptoJS.enc.Utf8.parse(tx
tUserName), key,
            {
                keySize: 128 / 8,
                iv: iv,
                mode: CryptoJS.mode.CBC,
                padding: CryptoJS.pad.Pkcs7
            });

            encrypted.push(encryptedlogin);
```

Обмін даними з серверною частиною реалізовано за допомогою бібліотеки axios [5]. Для звернення до програмного інтерфейсу (API), що на хостингу, нашою командою було прийняте рішення створити субдомен, а саме — https://api.funtab.com.ua/. Бібліотека axios [5] дає змогу створити окремий instance і надалі його неодноразово використовувати. Внаслідок цього, при потребі звернення до серверу, не потрібно кожний раз прописувати параметри необхідні для запиту, адже всі були вони записані при створенні instance axios.

```
const instance = axios.create({
    baseURL: 'https://api.funtab.com.ua/',
    headers: {
        "Content-Type": 'application/json'
    }
});
```

Нижче наведений приклад звернення до API:

```
export const registerAPI = {
    registration(data) {
        return
instance.post('/auth.php', data);
    }
}
```

Такі об’єкти, що мають у собі функції, які повертають результати post-запиту, прописані в одному js-файлі.

```
import { authAPI, unAuthAPI } from
"../api/api";
```

Ці об’єкти надалі імпортуються та використовуються в редюсерах, а вже через них потрібні дані передаються в відповідні компоненти [1].

```
export const setAuthVideos = (videos) => ({ type:
SET_AUTH_VIDEOS, videos });
export const getAuthVideos = () => (dispatch) => {
   dispatch(setAuthVideos(authAPI.getVideos()));
}
```

Для з’єднання компонента з необхідним редюсером застосовується функція connect з бібліотеки react-redux [3]. В цю функцію передається компонент, та зазначається, які дані потрібні з якого редюсера, а функція, в свою чергу, повертає новий компонент з необхідними даними у props-об’єкті.

```
let mapStateToProps = (state)
=> {
   return {
      isAuth: state.auth.isAuth,
      videos: state.auth.videos,
      blueChannels:
state.auth.blueChannels
   }
};

export default
connect(mapStateToProps,
   { getUnAuthVideos,
getAuthVideos }
)(Main);
```

Таким чином відбувається взаємодія з серверною частиною, та реагуванням на дані, які вона повертає.

Візуальну частину сайту реалізовано за допомогою бібліотеки Material UI [6], яка дозволяє створити дуже гнучкий інтерфейс.

Повернемося до знайомства авторизованого користувача з сайтом. Після реєстрації, автоматично створюється пустий канал, який можна модифікувати: вибрати картинку канала та його фону, загрузити відео. Однією з ідей нашої команди, була можливість надання користувачеві, при публікації потокового контенту, встановлення постеру, який буде відображатись, коли відео неактивне, але це не обов'язково. Всі перелічені дії в цьому абзаці супроводжуються збереженням файлів в папці public та зверненням до серверної частини, способом описаним вище, задля збереження нововведень в базу даних. Також користувач не обмежений одним каналом, їх може бути декілька.

Хочеться підкреслити, що командою була також запропонована ідея, яка була успішно реалізована на клієнтській частині проекту, а саме: виділення підписки та відео, якщо воно з цього каналу. Візуально це виглядає так (див. рис. 1):

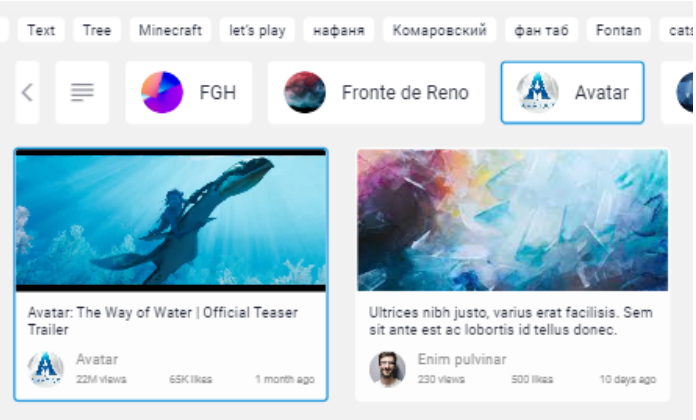


Рисунок 1 - Демонстрація нововведення

Демонструється це нововведення одразу при загрузці контентної стрічки на головній сторінці після авторизації. Дана функція полегшує користувачеві задачу виокремлення відеоконтенту, каналів на які він підписаний, серед великого потоку пропонуємих відео. А реалізовано це так:

```
getBlueChannels() {
        const subs =
subscriptions;
        const videos =
authVideoCards;
        let res = [];
        for (const card of videos)
{
            for (const sub of subs)
{
                if
(sub.channelName ===
card.channelName) {

res.push(sub.channelName);
                }
            }
        }

        return res;
    }
```

При зверненні до серверної частини та отриманню підписок та потокового контенту, проводиться перевірка на збіг каналів, якщо так, то назва цього каналу додається в масив, а сам масив передається через редюсери до компонентів, які відображають підписки та відеоконтент, а в цих компонентах вже проводиться перевірка на відповідність при якій відбувається блакитне виділення, або ні. Так це реалізовано для потокового контенту в компоненті Main:

```
{props.videos.map((card) =>
{
        isBlue = false;
        for (const b of
props.blueChannels) {
            if
(card.channelName == b) {
                isBlue = true;
            }
        }

        return <VideoCard
            isBlue={isBlue} ...
```

Всередині VideoCard:

```
return <MyCard key={props.id} sx={{borderColor:
isBlue ? 'primary.main' : 'transparent'}} > …
```

З підписками реалізація аналогічна.

Тепер про авторизованість. Коли користувач здійснив вхід у систему, він може підписуватися на канали інших користувачів, отримувати повідомлення, ставити оцінку (like/dislike) відеоконтенту та коментувати його. Ще немало корисним є можливість групування потокового контенту, наприклад, в плейлисти, історія переглядів. Ці всі дії, а також зацікавленість пенними тегами чи категоріями фіксуються через звернення до програмного інтерфейсу, для того, щоб в майбутньому сформувати головну стрічку відео, підбір тегів, які будуть актуальні саме для конкретного користувача.

Щодо функцій, які доступні всім користувачам. Генерація, демонстрація головної стрічки, перегляд каналів користувачів, незалежно від входу в систему. Цей аспект впливає на персоналізованість та надання більших прав. Також доступна фільтрація контенту тегами та категоріями. Доступний пошук відео. Хочу підкреслити, що ця функція була розширена голосовим вводом тексту, поки що тільки англійська мова, а в перспективі планується ще декілька мов. Реалізована за допомогою додатково встановленого npm-пакету ‘react-speech-recognition’ [8]. Приклад використання:

```
const {
    transcript,
    listening,
    resetTranscript,
    browserSupportsSpeechRecognition
  } = useSpeechRecognition();

  const [isActiveSpeech, setActiveSpeech]
= useState(null);
  useEffect(() => {
    if (isActiveSpeech) {
      SpeechRecognition.startListening();//{
language: 'uk-UA', continuous: true }

    } else if (isActiveSpeech === false) {
props.toggleSearchTextValue(transcript);
      setTextValue(transcript);
      SpeechRecognition.stopListening();

      setActiveSpeech(null);
    }
  }, [isActiveSpeech]);
```

А однією з головних функцій даного сайту є відтворення відео. Тому був розроблений власний відео-плеєр. Але, так як всі файли в тому числі і відео розміщені в public папці сайту, для відтворення конкретного відео потрібна його назва в цій папці, а назва зберігається в базі даних, тому робиться запит до програмного інтерфейсу, задля отримання, як це було описано раніше.

Як вже зазначалось вище, з повним кодом проекту можна ознайомитись у репозиторії https://VladislavaVG@bitbucket.org/VladislavaVG/funtab.git.

РОЗДІЛ 3
РЕЗУЛЬТАТИ ВИПРОБУВАНЬ

В результаті збірки програмного коду, всіх пакетів та залежностей клієнтської частини командою “npm build” являється папка build, яку можна загрузити на хостинг та відкрити в будь-якому браузері. Головна серія випробувань, звіт з якої наведено далі, проведена в браузері Chrome.

Знайомство нового користувача з сайтом починається з головної сторінки. Зовнішній вигляд наведено на рис. 2.

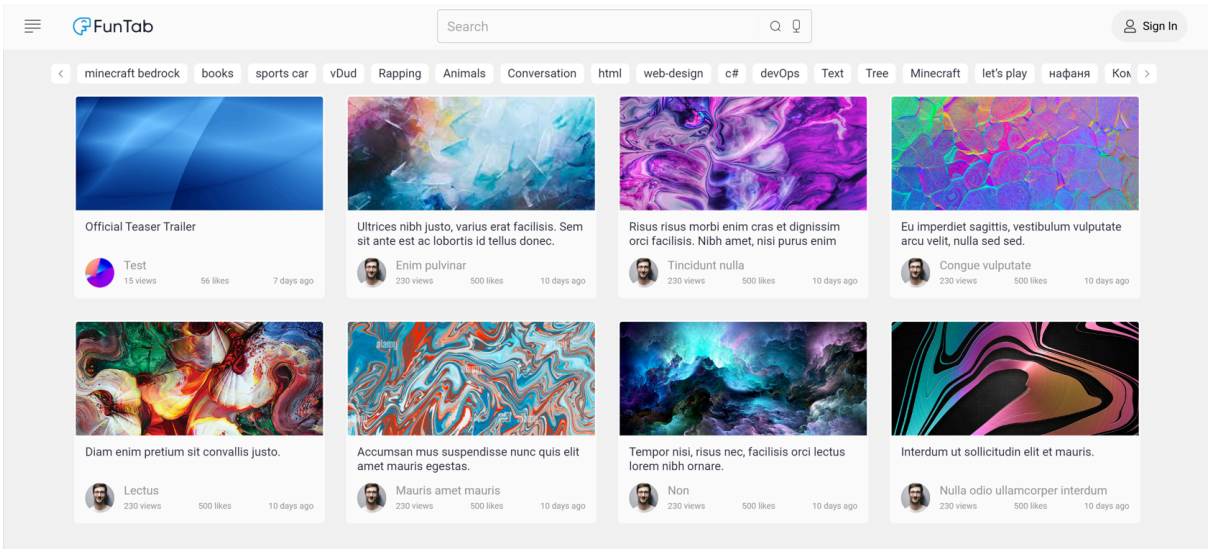


Рисунок 2 - Головна сторінка неавторизованого користувача

Але, задля отримання більших можливостей(які детально описані в минулому розділі), майбутньому учаснику системи потрібно зареєструватися. Під час реєстрації здійснюється перевірка засобів комунікації - на електронну пошту надсилається код підтвердження, який має бути введений для завершення етапу реєстрації. Зовнішній вигляд вікон реєстрації та входу наведено на рис. 3.

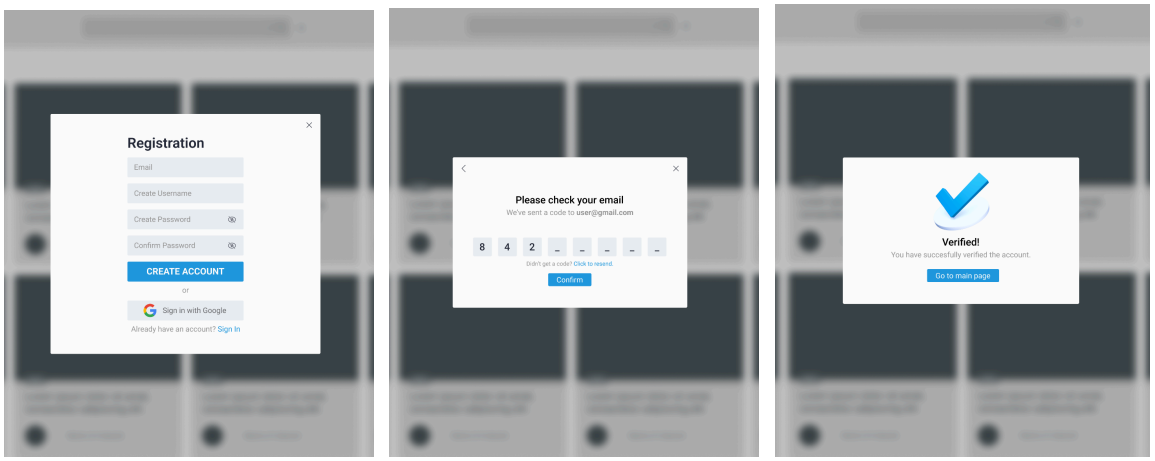


Рисунок 3 - Реєстрація користувача, підтвердження пошти та вхід

Якщо користувач раніше проходив реєстрацію, то він може увійти до системи за допомогою персональних даних, зазначених при реєстрації. Зовнішній вигляд вікна авторизації, зображений на рис. 4.

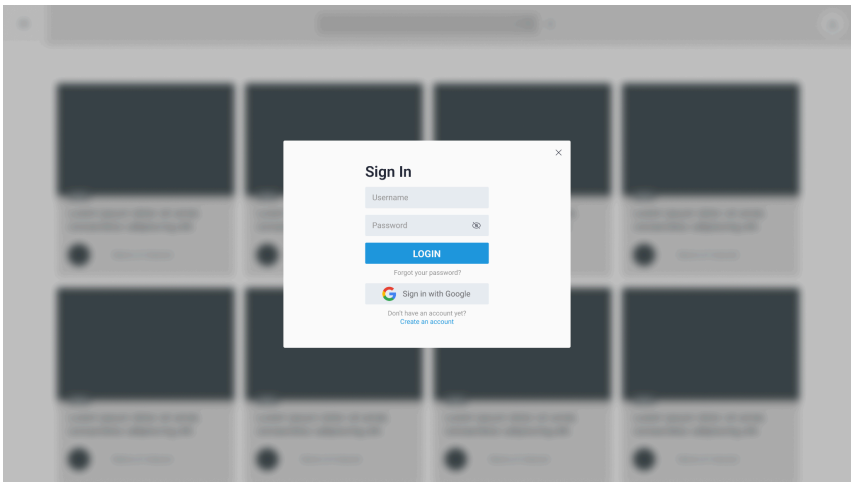
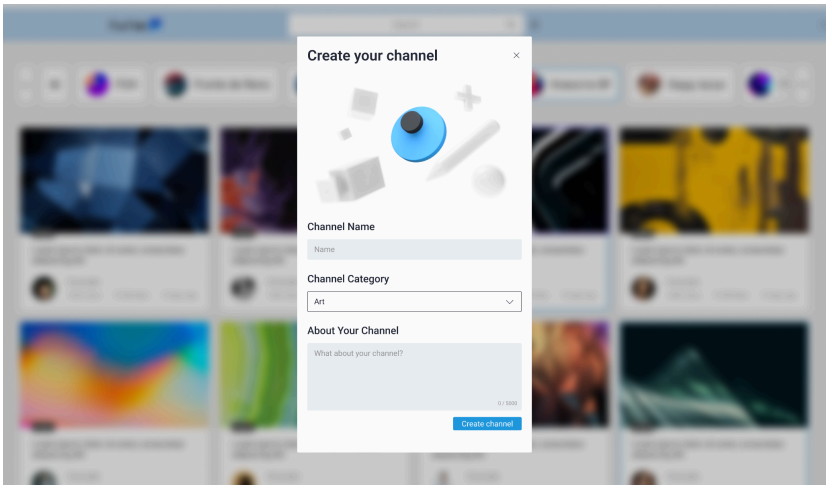


Рисунок 4 - Авторизація користувача

Після входу до системи користувач отримує доступ до таких функцій системи:

Створення каналу. Зовнішній вигляд цієї сторінки, зображений на рис. 5.



.

Рисунок 5 - Створення каналу

Підписки на канали. Зовнішній вигляд цієї сторінки, зображений на рис. 6

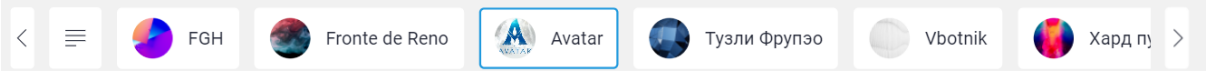


Рисунок 6 - Відображення підписок

Фільтрування контенту через теги, категорії. Зовнішній вигляд цієї сторінки, зображений на рис. 7

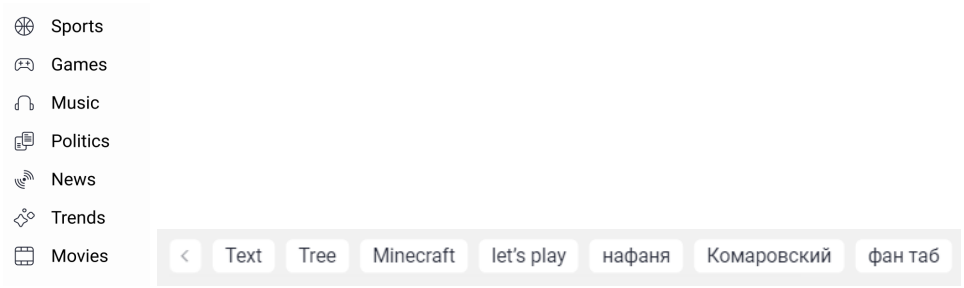


Рисунок 7 - Категорії та теги

Оцінювання, відтворення, коментування відеоконтенту. Зовнішній вигляд цієї сторінки, зображений на рис. 8

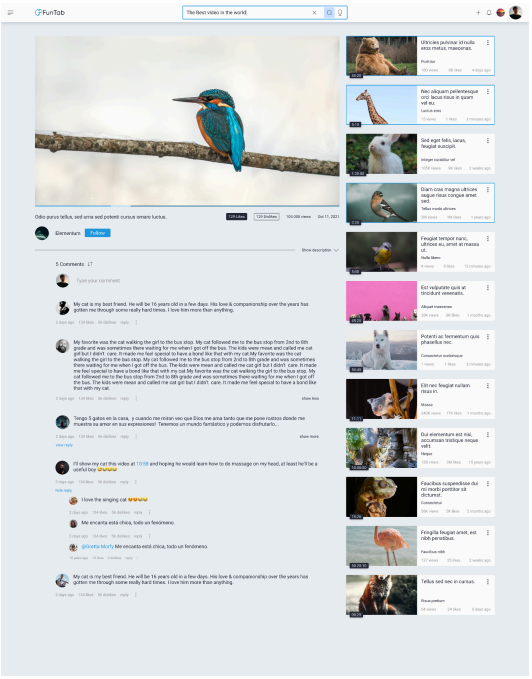


Рисунок 8 - Відкрите відео

Групування. Зовнішній вигляд цієї сторінки, зображений на рис. 9

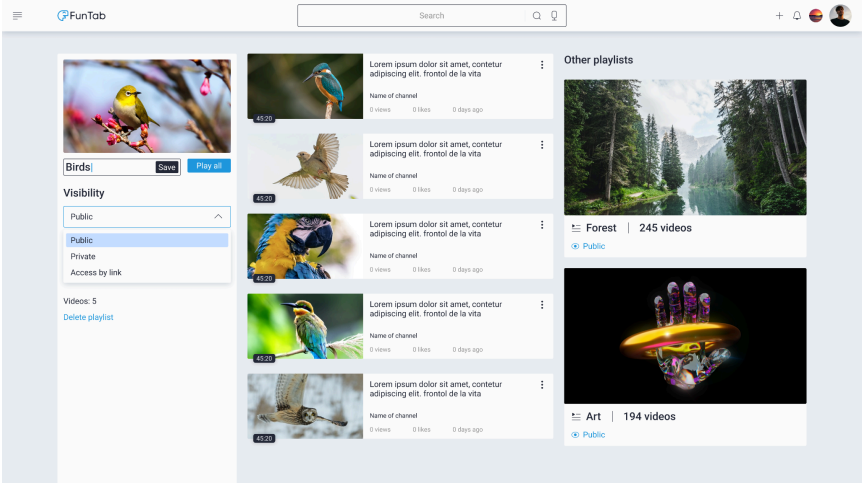


Рисунок 9 - Сторінка плей-листу

Пошук відео. Зовнішній вигляд цієї сторінки, зображений на рис. 10

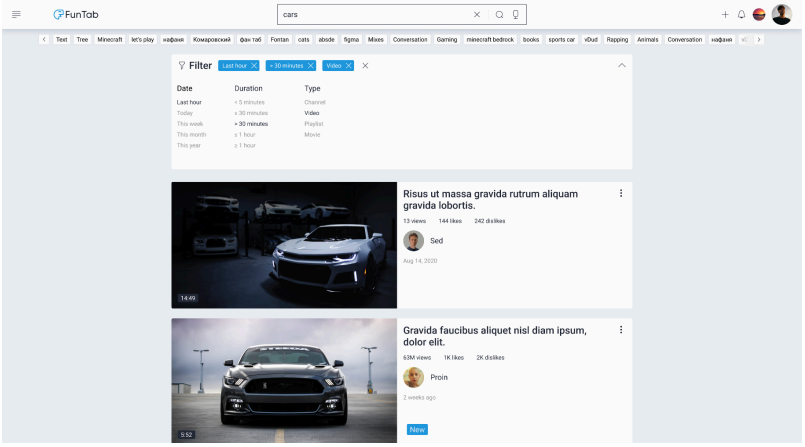


Рисунок 10 - Сторінка пошуку відео

Всі дії фіксуються в базі даних через запити до програмного інтерфейсу серверної частини. Подальша комунікація здійснюється через особистий кабінет каналу користувача

Через кабінет користувач має змогу коригувати персональні дані, модифікувати канал, налаштовувати приватність і т.д. Зовнішній вигляд цієї сторінки, зображений на рис. 11

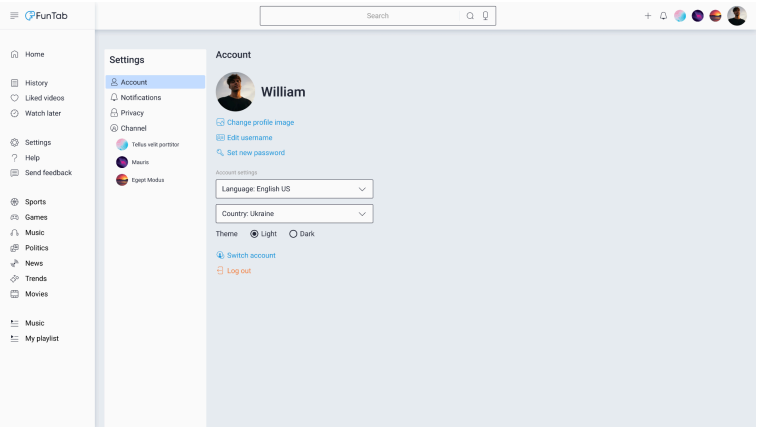


Рисунок 12 - Сторінка налаштувань

У підсумку, всі функції клієнтської частини, що були регламентовані технічним завданням було реалізовано та випробувано. Поточна версія сайту, що реалізує клієнтську частину, доступна у репозиторії за цим посиланням: https://VladislavaVG@bitbucket.org/VladislavaVG/funtab.git.

А сам сайт можете переглянути за цим посиланням: https://funtab.com.ua/.

ВИСНОВКИ

Проведено аналіз інформаційних систем, що наразі є найбільш актуальними для популяризації потокового контенту на велику аудиторію, з можливістю комерційного призначення, виявлено, що суттєве поширення займають системи керування потоковим контентом. Саме для таких систем відео мають основне значення. За приклад для реалізації було взято YouTube.

Визначено програмні засоби (інтерфейси API), які потрібні для функціонування проекту, перевагу надано засобам: GoogleLogin API, SpeechRecognition API. Дані засоби було вжито при розробці: ReactJS, Redux, React Redux, Redux Form, Material UI, axios.

Спроектовано, реалізовано та випробувано систему керування потоковим контентом. Система розроблена за архітектурою «Progressive Web Application» (PWA) та складається з трьох частин, розміщених за адресами:

- «Mockup» https://www.figma.com/file/NWgcgIlTsZgB1ugP0pxNNm/Design-FunTab?node-id=947%3A6483,
- «Backend» https://funtab.com.ua/ (на хостингу),
- «Frontend» https://VladislavaVG@bitbucket.org/VladislavaVG/funtab.git.

У систему впроваджено засоби покращення інформаційної безпеки: додаткове шифрування даних, що передаються комунікаційним каналом, за алгоритмом AES-128. Також вжито засоби підтвердження персональних даних користувача, зокрема, верифікаційний код для перевірки електронної пошти. Це ускладнює зловмисникам вживання підроблених (невласних) даних для дискредитації системи.

Систему було розміщено на постійному хостингу та випробувано на різних браузерах, зокрема мобільних. Результати випробувань підтвердили загальну працездатність створеної системи.

Перспективи подальших досліджень вбачаються у розвинені таких функцій системи як ведення прямих ефірів, впровадження реклами, монетизація потокового контенту.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. React - JavaScript-бібліотека для створення користувацьких інтерфейсів. URL: https://uk.reactjs.org/
2. Redux — A Predictable State Container for JS Apps. URL: https://redux.js.org/
3. React Redux - Official React bindings for Redux. URL: https://react-redux.js.org/
4. Redux Form. URL: https://redux-form.com/8.3.0/
5. Axios — Promise based HTTP client for the browser and node.js. URL: https://axios-http.com/docs/intro
6. Material UI Design. URL: https://mui.com/
7. React Google Login. URL: https://www.npmjs.com/package/react-google-login
8. Speech Recognition. URL: https://www.npmjs.com/package/react-speech-recognition