

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

ДЛЯ ОБРОБКИ ВІДЕОПОТОКІВ

VIDEO STREAM PROCESSING SOFTWARE

Виконав(ла): студент(ка) 4 курсу, групи КНТ-122

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

РІЗНИК Д.Ю.

(ПРИЗВИЩЕ та ініціали)

Керівник КОЛПАКОВА Т.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент КОЦУР М.І.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

РІЗНИКА Дмитра Юрійовича

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення для обробки відеопотоків.
Video Stream Processing Software

керівник проєкту (роботи) к.т.н., доцент, КОЛПАКОВА Тетяна Олексіївна,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 07 ” квітня 2026 року № 139

2. Строк подання студентом проєкту (роботи) 03 червня 2026 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне
завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) _____

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	КОЛПАКОВА Т.О., доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю., старший викладач		

7. Дата видачі завдання “07” квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Дмитро РІЗНИК
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Тетяна КОЛПАКОВА
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
90 с., 3 табл., 26 рис., 3 дод., 19 джерел.

JAVASCRIPT, REACT, ВІДЕОПОТІК, МОВА ПРОГРАМУВАННЯ,
ПРОГРАМНА БІБЛІОТЕКА, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Об'єкт дослідження – процес розробки програмного забезпечення для обробки відеопотоків.

Предмет дослідження – програмні засоби для підтримки процесу обробки відеопотоків.

Мета роботи – розробка програмного забезпечення для обробки відеопотоків.

Матеріали, методи та технічні засоби: мова програмування JavaScript, бібліотека React.

Результати. Розроблено проєктні рішення для створення програмного забезпечення для обробки відеопотоків. Створено програмне забезпечення для підтримки процесу обробки відеопотоків. Проведено тестування розробленої програмної системи для підтримки процесу обробки відеопотоків.

Висновки. Мету роботи досягнуто. Розроблено програмне забезпечення для обробки відеопотоків за допомогою мови програмування JavaScript та бібліотеки React.

Галузь використання – обробка відеопотоків.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 90 pages, 3 tables, 26 figures, 3 appendixes, 19 sources.

JAVASCRIPT, REACT, VIDEO STREAM, PROGRAMMING LANGUAGE, SOFTWARE LIBRARY, SOFTWARE.

The object of research is the process of developing software for video streams processing.

The subject of the research is software for supporting the the process of processing video streams.

The purpose of this work is to develop the software to support the video streams processing.

Materials, methods and technical tools: JavaScript programming language, React library.

Results. Project solutions for the creation of software for video streams processing has been designed. The software to support the process of video streams processing has been developed. The developed software system for video streams processing has been tested.

Conclusions. The goal of the work has been achieved. The software tools for video streams processing using the JavaScript programming language and the React library have been developed.

Scope of use – video streams processing.

ЗМІСТ

	С.
Перелік скорочень та умовних познач	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Програмне забезпечення для обробки відеопотоків	11
1.2 Аналіз програмного забезпечення для обробки відеопотоків	21
1.3 Висновки за розділом 1	30
2 Матеріали і методи	32
2.1 Вибір мови програмування	32
2.2 Вибір бібліотеки для створення програмного забезпечення для обробки відеопотоків	35
2.3 Висновки за розділом 2	38
3 Опис програми	39
3.1 Структура програмного забезпечення для обробки відеопотоків	39
3.2 Опис модулів програмного забезпечення для обробки відеопотоків	41
3.3 Функціонування програмного забезпечення для обробки відеопотоків	43
3.4 Проєктування інтерфейсу програмного забезпечення для обробки відеопотоків	46
3.5 Висновки за розділом 3	48
4 Експлуатація, тестування та експериментальне дослідження програми.....	50
4.1 Призначення й умови застосування програми	50
4.2 Характеристики програми для обробки відеопотоків	51
4.3 Інструкція по експлуатації програми.....	52
4.3.1 Звернення до програми	52
4.3.2 Вхідні й вихідні дані	52
4.3.3 Повідомлення.....	53
4.4 Виконання програмного забезпечення для обробки відеопотоків	53
4.5 Тестування програмного забезпечення для обробки відеопотоків	56
4.6 Висновки за розділом 4	56

Висновки	58
Перелік джерел посилання	61
Додаток А Технічне завдання	63
Додаток Б Фрагмент тексту програми	67
Додаток В Слайди презентації	84

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – Application Programming Interface;

RTC – Real Time Communication;

БД – база даних;

ПЗ – програмне забезпечення.

ВСТУП

Актуальність програмного забезпечення для обробки відеопотоків зумовлена стрімким зростанням обсягів відеоінформації у світі. На сьогоднішній день відео займає суттєву частку глобального Інтернет трафіку, і це зростання продовжується завдяки поширенню відеоспостереження, онлайн сервісів передачі відеоінформації, платформ для відеоконференцій та мобільних застосунків. Усе це створює значний попит на програмні інструменти, здатні ефективно обробляти великі обсяги відеоданих, зокрема, у режимі реального часу [1]-[3].

Особливо важливою така обробка є в критичних сферах, де затримка у прийнятті рішень може мати серйозні наслідки. Це стосується систем безпеки, моніторингу громадських місць, інтелектуального відеоспостереження, автономного транспорту, дронів і телемедицини. У цих сферах програмне забезпечення повинно забезпечувати швидке розпізнавання подій, виявлення об'єктів і прийняття рішень на основі відеоаналітики [4], [5].

Проте деяке програмне забезпечення для обробки відеопотоків має низку недоліків, які слід враховувати під час його впровадження та використання. Однією з головних проблем є висока вимогливість до апаратного забезпечення. Обробка відео, особливо в реальному часі, потребує значних обчислювальних ресурсів, зокрема продуктивних процесорів, графічних прискорювачів і великої кількості оперативної пам'яті. Це підвищує загальну вартість системи й ускладнює її масштабування, особливо для малих підприємств або у випадках обробки відео на периферійних пристроях із обмеженими ресурсами. Ще одним суттєвим недоліком є складність у налаштуванні та інтеграції деяких програмних засобів для обробки відеопотоків. Багато програмних рішень потребують технічних знань для інсталяції, конфігурації та обслуговування. У деяких випадках потрібна участь спеціалістів із машинного навчання, комп'ютерного зору або мережевої

безпеки, що обмежує доступність цих технологій для непрофесійних користувачів або невеликих команд [1]-[6].

Тому актуальною є розробка програмного забезпечення для обробки відеопотоків. У дипломній кваліфікаційній роботі бакалавра розв'язується актуальне завдання розробки програмного забезпечення для обробки відеопотоків.

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для обробки відеопотоків;
- здійснити проектування програмного забезпечення для обробки відеопотоків;
- створити програмне забезпечення для обробки відеопотоків;
- виконати тестування розробленого програмного забезпечення для обробки відеопотоків.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Програмне забезпечення для обробки відеопотоків

Відеопотік передає безпосередньо через мережу фрагменти мультимедійних даних, що дає змогу почати перегляд відразу ж після ініціалізації відтворення, без потреби чекати на завантаження цілого файлу. Такий підхід відкриває миттєвий доступ до широкого спектра мультимедійного контенту, зокрема, кінокартин, звукових записів, телевізійних шоу або прямих трансляцій, і дозволяє переглядати їх на різноманітних пристроях, починаючи з мобільних пристроїв і закінчуючи телевізорами з підтримкою Інтернету [1].

Механізм роботи ґрунтується на розподілі великих файлів на серію менших блоків даних. Коли ініціюється відтворення, ці блоки надходять до пристрою користувача в режимі реального часу, і відтворення починається без затримок, характерних для традиційного завантаження. Для пересилання цих блоків задіюються різні мережеві протоколи, кожен із яких має власні переваги та обмеження [1].

Перший із застосовуваних методів передбачає пересилання без встановлення з'єднання та без контролю порядку надходження блоків. Завдяки цьому дані рухаються з мінімальними затримками й мережевими накладними витратами, проте втрата окремих фрагментів не компенсується повторною передачею. Такий спосіб використовується в тих випадках, коли важливіше уникнути затримок, навіть якщо підприємство відтворення іноді позначається на якості через пропуски чи артефакти [1].

Інший метод передбачає встановлення динамічного з'єднання із перевіркою цілісності кожного блоку та можливістю повторної передачі загублених даних. Цей підхід відрізняється адаптацією до стану мережі: у разі перевантаження швидкість передавання скорочується, а за покращення умов автоматично зростає, що, хоч і уповільнює початкову доставку, забезпечує плавне та високоякісне відтворення без пропусків і збоїв [1].

Нині розробляються гібридні рішення, які поєднують швидкість і легкість першого методу з надійністю другого, прагнучи мінімізувати затримки та втрати даних одночасно. Розуміння здатностей кожного з цих способів має ключове значення для тих, хто створює інструменти для обробки відеопотоків, адже вибір між високою швидкістю та гранульованою надійністю визначає досвід кінцевого глядача [1], [2].

Існують різні способи передачі відео контенту через мережу, які умовно поділяються на трансляції в реальному часі та перегляд за запитом. Трансляція в реальному часі дає можливість спостерігати події у момент їхнього відбування, що широко використовується під час спортивних матчів, концертів, новин або онлайн-занять. Цей метод відзначається інтерактивністю, оскільки глядачі можуть брати участь у чатах, залишати коментарі та реагувати на трансляцію, що робить процес перегляду більш залученим та живим. Крім того, така форма відеопотоків часто супроводжується функціями монетизації, зокрема рекламою, підписками чи пожертвами [1], [2].

Інший підхід полягає у наданні доступу до заздалегідь записаних відеоматеріалів, які можна переглядати у будь-який зручний час. Цей режим характеризується можливістю вільно керувати відтворенням: ставити на паузу, перемотувати або повторно переглядати потрібні фрагменти, що забезпечує індивідуальний підхід до контенту. Різноманітні моделі організації таких сервісів передбачають оплату підписки, оплату за окремі перегляди або безкоштовний доступ з рекламними вставками [1], [2].

Для розробників та бізнесу використання потокового відео відкриває численні переваги. Серед них – можливість реалізації різних варіантів монетизації, що охоплюють як підписки, так і рекламу або платний перегляд, а також інтеграція з різними платіжними системами для зручності користувачів. Сучасні технологічні рішення, такі як WebRTC, адаптивне регулювання якості потоку і хмарні обчислення, дозволяють створювати ефективні, масштабовані системи, які відповідають найновішим стандартам. Ефективне розподілення ресурсів та використання мереж доставки контенту допомагає мінімізувати

витрати на серверне обладнання. Потокові платформи також збирають велику кількість даних про користувацьку поведінку, що дає змогу аналізувати вподобання аудиторії і покращувати якість сервісу. Крім того, модульна архітектура програмних компонентів дозволяє повторно використовувати код у різних проектах, що прискорює розробку і підтримку [1], [2].

Водночас організація потокового відео пов'язана з певними викликами. Створення надійної інфраструктури потребує спеціалізованих знань у сфері мереж, серверних технологій і доставки контенту, що робить початковий етап доволі складним і трудомістким. Забезпечення безпеки відеоматеріалів і даних глядачів вимагає впровадження комплексних заходів, включно з цифровим управлінням правами та автентифікацією, а також дотриманням регуляторних норм. Тестування і контроль якості у потокових сервісах ускладнюються необхідністю враховувати різноманітність пристроїв, варіативність мережевих умов і потенційні виняткові ситуації. Швидкий розвиток технологій у цій галузі змушує регулярно оновлювати продукти, що іноді призводить до накопичення технічного боргу, керування яким стає дедалі складнішим. Паралельно, оптимізація продуктивності вимагає глибокого розуміння роботи кодеків, мережевого трафіку та буферизації, а також постійної адаптації системи для збалансування якості та використання ресурсів [1], [2].

Потокова передача відео стала невід'ємною складовою сучасних цифрових сервісів, що вимагає застосування специфічних технологій для її організації. Використання HTML5 дозволяє безпосередньо вбудовувати відео контент у вебсторінки без необхідності сторонніх плагінів, що робить відео сумісним з широким спектром браузерів і пристроїв. При цьому адаптивні алгоритми передавання якості відео автоматично підлаштовують потік під поточну пропускну здатність мережі користувача, забезпечуючи плавний перегляд без зупинок [1], [2].

Програмні системи для керування відео забезпечують зручність у публікації та розповсюдженні матеріалів, спрощуючи інтеграцію контенту в різноманітні платформи. Для ефективної обробки відеоданих необхідно

враховувати особливості кодеків та контейнерів, які впливають на якість і розмір відеофайлів, а також сумісність із пристроями відтворення [1], [2].

Правильна побудова потокової інфраструктури передбачає знання протоколів, що регламентують передачу відео в мережі, адже від цього залежить стабільність та швидкість доставки контенту. Спеціалізовані бібліотеки для відтворення відео полегшують імплементацію додаткових функцій, таких як субтитри або збір аналітики, що підвищує зручність користувачів [2], [3].

У виборі технологічного підходу враховуються специфічні потреби платформи. Для кросплатформених мобільних додатків часто застосовують рішення, які дозволяють писати код, що працює і на Android, і на iOS, зберігаючи продуктивність і скорочуючи час розробки. Програми, орієнтовані виключно на одну платформу, можуть використовувати програмні засоби для максимальної оптимізації, що особливо важливо в проектах з високими вимогами до якості відео і інтерактивності. Для вебзастосунків популярними є сучасні фреймворки, здатні підтримувати передачу даних у реальному часі через WebRTC [2], [3].

На серверній стороні застосовуються рішення, що поєднують продуктивність і масштабованість, здатні обробляти велику кількість одночасних трансляцій з мінімальною затримкою. Комбінації певних мов програмування та мультимедійних фреймворків оптимізують роботу з відеопотоками і зменшують навантаження на сервери [2], [3].

Програмні продукти для організації потокового відео виконують функції кодування, трансляції і запису контенту, дозволяючи управляти потоками як новачкам, так і досвідченим фахівцям. Такі програми сприяють створенню онлайн трансляцій та відео бібліотек [3], [4].

Додатки для потокового телебачення надають змогу переглядати різноманітні передачі через Інтернет, пропонуючи гнучкий і зручний доступ до контенту, що може транслюватися в режимі реального часу або бути доступним за запитом.

Основна відмінність між потоковою передачею та завантаженням полягає в способі доставки медіа: у першому випадку відео відтворюється безпосередньо під час отримання даних, у другому — весь файл завантажується на пристрій перед переглядом, що дозволяє користувачеві дивитись контент без підключення до мережі [3], [4].

Обробка відеопотоків передбачає передачу мультимедійного контенту через Інтернет без необхідності завантажувати файл повністю на пристрій. Відео чи аудіо надходить у вигляді послідовних фрагментів, які відтворюються відразу після отримання. Такий підхід дозволяє уникнути затримок, пов'язаних із повним завантаженням файлу перед переглядом або прослуховуванням [3], [4].

Різні платформи використовують цей принцип для доставки контенту користувачам. Наприклад, популярні сервіси дозволяють миттєво отримувати доступ до фільмів, музики або трансляцій, не зберігаючи їх цілком на локальному пристрої. Існують різноманітні способи організації потокового передавання, які залежать від типу медіа, методів доставки та вимог до споживання контенту [3], [4].

Один із варіантів передбачає можливість вибору моменту перегляду чи прослуховування, без необхідності чекати на повне завантаження. Інший підхід полягає у трансляції подій у режимі реального часу, що дає змогу стежити за розвитком подій без затримок. Також застосовується методика, за якою файли поступово завантажуються під час відтворення, іноді дозволяючи зберігати їх частково або повністю на пристрої. Для інтерактивних комунікацій, таких як відео дзвінки або онлайн ігри, використовують технології з мінімальними затримками, що забезпечують майже миттєву передачу даних [3], [4].

Аудіоконтент транслюється окремо, із спеціалізованими платформами, які забезпечують безперервне відтворення музики, радіо чи подкастів. Сучасні рішення часто оснащені механізмами, що автоматично регулюють якість

потоків залежно від пропускної здатності Інтернет з'єднання користувача, завдяки чому забезпечується стабільний перегляд без переривань [3], [4].

Загалом, технологія потокового передавання базується на передачі даних частинами, що дозволяє починати відтворення практично одразу, без тривалого очікування. Це досягається за рахунок безперервної комунікації між сервером, який передає мультимедіа, і пристроєм користувача, який приймає і відтворює отримані сегменти контенту [3], [4].

Під час передачі відеосигналу в режимі реального часу дані надсилаються на пристрій користувача поступово, без попереднього завантаження повного файлу. Передавання здійснюється через мережеві канали за допомогою спеціальних програмних засобів, що забезпечують безперервне оновлення відеопотоку. Весь процес розгортається у фоновому режимі: відео розбивається на послідовні частини, які надходять з віддаленого джерела поетапно. Кожен сегмент тимчасово зберігається у внутрішній пам'яті програвача, де формується буфер, що згладжує коливання швидкості з'єднання [4], [5].

Використання кодеків дозволяє стискати та розпаковувати відеоінформацію, завдяки чому забезпечується ефективна передача і відтворення. Усе це відбувається автоматично — дані надходять, розпізнаються і виводяться на екран без участі користувача. При зменшенні пропускної здатності мережі система динамічно адаптує якість зображення, підтримуючи безперервність перегляду. Спеціалізовані служби доставки контенту допомагають скоротити затримки, передаючи відео з серверів, що розташовані географічно ближче до кінцевого користувача [4], [5].

Після припинення перегляду передача зупиняється, і тимчасові дані з пам'яті пристрою поступово видаляються. У разі повторного запуску того самого матеріалу, можливо, частину фрагментів буде отримано швидше завдяки проміжному зберіганню. Увесь процес відбувається завдяки взаємодії мережевих технологій та програмного забезпечення, призначеного для обробки поточних даних у реальному часі [4], [5].

Потокове передавання — це сукупність методів і рішень, які дозволяють безперервно передавати мультимедійний вміст через мережу, забезпечуючи стабільне відтворення без необхідності попереднього завантаження всього файлу [4], [5].

Щоб зробити передавання мультимедійних даних можливим, попередньо виконується стискання оригінальних відео- та аудіоматеріалів. Це дає змогу зменшити обсяг переданих даних і пришвидшити доставку, зберігаючи при цьому прийнятну якість. Декодування стисненого сигналу відбувається на стороні користувача за допомогою відповідного програмного забезпечення [4], [5].

Дані медіафайли передаються згідно з певними правилами, які визначають, яким чином інформація потрапляє від джерела до пристрою користувача. Ці механізми передбачають поділ вмісту на окремі частини, адаптацію до умов мережі та динамічну зміну якості під час перегляду. Рішення, що базуються на цих принципах, активно використовуються в мобільних застосунках, на відео сервісах та під час відео конференцій [4], [5].

Для підвищення швидкості доступу до вмісту та зменшення навантаження на основні сервери застосовуються географічно розподілені вузли, які дублюють дані та передають їх користувачам з найближчого джерела. Завдяки цьому знижується час очікування та зменшується ймовірність збоїв під час передавання [4], [5].

Якість зображення й звуку змінюється в реальному часі залежно від можливостей Інтернет з'єднання. При зменшенні пропускної здатності система автоматично переходить на нижчу роздільну здатність, щоб забезпечити плавне відтворення без зупинок. Такий підхід дозволяє підтримувати стабільну роботу сервісу незалежно від зовнішніх умов [4], [5].

Для запобігання несанкціонованому доступу впроваджуються механізми контролю, що обмежують відтворення лише для дозволених користувачів. Захист реалізується шляхом шифрування потоку, який може бути розкритий лише в разі наявності відповідного дозволу [4], [5].

Підготовка контенту для потокової передачі передбачає не лише початкове стискання, а й подальше перетворення в різні формати та якості, що дає змогу адаптувати вміст до характеристик різних пристроїв і швидкості мережі. Прийом, обробка й показ мультимедійного сигналу здійснюються спеціальними програмами, які працюють на кінцевих пристроях. Вони відповідають за відображення, буферизацію та налаштування параметрів трансляції відповідно до реального стану мережевого середовища [4], [5].

Функціонування поточкових сервісів підтримується за рахунок віддалених обчислювальних систем, які забезпечують зберігання, обробку й масштабування обсягу обслуговування. Такі системи дозволяють підтримувати стабільність під час різкого зростання попиту, зокрема під час масових трансляцій або пікових навантажень [4], [5].

У сфері потокового передавання використовуються два основні мережеві протоколи, кожен із яких краще підходить для певних умов передавання аудіовізуального контенту через Інтернет. Один із них спрямований на максимальну точність і послідовність, інший — на мінімізацію затримок, навіть ціною втрати частини даних [5], [6].

У ситуаціях, де потрібна повна цілісність даних і допустима незначна затримка при відтворенні, зазвичай обирається варіант, орієнтований на гарантування доставки інформації. Такі сервіси, як онлайн кінотеатри або музичні платформи, використовують методи, що дозволяють впевнено передавати дані, навіть якщо для цього потрібно повторно надсилати втрачені фрагменти. Це дозволяє зберігати якість трансляції, забезпечуючи її безперервність та відсутність помилок [5], [6].

Інший підхід зазвичай застосовується там, де пріоритетом є швидкість реакції та безперервність потоку в реальному часі. У таких випадках втрата деякої частини даних вважається допустимою, якщо це дозволяє зменшити затримку між джерелом сигналу та кінцевим користувачем. Це особливо актуально для трансляцій наживо, онлайн ігрових платформ або відео

конференцій. Плавність відтворення в таких випадках важливіша за абсолютну точність доставки [5], [6].

Сучасні технології також передбачають комбінування підходів, де вводяться спеціальні транспортні протоколи, здатні поєднувати переваги швидкості з основними засобами контролю якості. Таким чином досягається компроміс між швидкістю передавання і стабільністю [5], [6].

Потокове передавання суттєво відрізняється від традиційного способу отримання контенту через збереження файлів. У першому випадку інформація надходить частинами у процесі її перегляду, і доступ до неї можливий тільки при активному підключенні до мережі. Після завершення сесії перегляду дані більше не зберігаються на пристрої. У другому — весь обсяг контенту передається та зберігається локально, що дозволяє звертатися до нього без необхідності повторного доступу до джерела [5], [6].

Технологічно трансляції в реальному часі реалізуються шляхом поетапної обробки даних — від захоплення події до її безпосереднього відтворення на екранах глядачів. Відеосигнал обробляється технічними засобами, стискається до оптимального об'єму, передається через мережу із застосуванням відповідного протоколу, після чого транслюється через сумісні системи з урахуванням пропускну здатності підключення кожного користувача. Додаткові алгоритми дозволяють адаптувати якість потоку до умов перегляду, що робить такий тип передавання ефективним навіть при обмежених ресурсах мережі [5], [6].

Очікується, що трансформація потокового передавання надалі залежатиме від технічного вдосконалення та змін у медіаспоживанні. Високоточна передача зображення стане більш звичною завдяки новим засобам зв'язку та обробки відео. Засоби взаємодії з контентом змінюватимуть сприйняття — через можливість безпосередньої участі у трансляціях, вибору ракурсів, а також впровадження елементів віртуального простору та доповненої реальності [5], [6].

Налаштування під глядача поступово удосконалюється завдяки аналізу даних та інтелектуальним системам, що дозволяє точніше враховувати індивідуальні смаки та звички. Зниження затримок у передаванні даних відкриває шлях до ще більшої реалістичності трансляцій — у випадках, коли моментальність є ключовою [5], [6].

Інтеграція комерційних можливостей у прямі трансляції змінює спосіб споживання не лише контенту, а й товарів, створюючи нові формати цифрової взаємодії. Розширення мовного доступу до контенту можливе завдяки інструментам автоматичного перекладу, що підвищує глобальну охоплюваність. При цьому посилюється інтерес до технологій, здатних працювати з меншими витратами енергії, що відповідає потребам сталого цифрового середовища [5], [6].

Отже, програмне забезпечення для обробки відеопотоків — це інструмент, який забезпечує отримання, зміну, аналіз або передачу відео в режимі реального часу. Таке ПЗ використовується для роботи з відео, що транслюється безперервно, наприклад, з камер спостереження, під час відео конференцій або онлайн трансляцій. Його завданням є забезпечення стабільної обробки зображення без затримок, покращення якості відео, стиснення даних, вставка графіки або тексту, адаптація до різних мережових умов, а також передача відео на інші пристрої чи платформи. Такі системи часто використовують у телебаченні, безпеці, освіті та розвагах [5], [6].

Визначено, що відеопотік передає безпосередньо через мережу фрагменти мультимедійних даних, що дає змогу почати перегляд відразу ж після ініціалізації відтворення, без потреби чекати на завантаження цілого файлу. Програмне забезпечення для обробки відеопотоків являє собою сукупність програмних рішень, які дозволяють приймати, аналізувати, редагувати, трансформувати або передавати відео в режимі реального часу або з мінімальною затримкою. Основна мета такого ПЗ полягає у забезпеченні безперервної роботи з відеопотоком, тобто потоком кадрів, що надходять у реальному часі, зберігаючи його якість та синхронізацію.

1.2 Аналіз програмного забезпечення для обробки відеопотоків

Проаналізуємо програмне забезпечення для обробки відеопотоків [7]-[15].

Програмне забезпечення vMix (рис. 1.1) слугує потужним інструментом для обробки відеопотоків у реальному часі, орієнтованим на створення та управління високоякісними трансляціями. Завдяки широкій підтримці відеоформатів та гнучкому опрацюванню вхідного сигналу, воно забезпечує плавну організацію поточкових трансляцій із різноманітних джерел, включаючи камери, відеофайли, презентації, зовнішні пристрої захоплення та мультимедійний контент [7]-[10], [13].



Рисунок 1.1 – Програмне забезпечення vMix [13]

Програмне забезпечення vMix підтримує з'єднання із соціальними платформами для прямого мовлення, дозволяє працювати з декількома рівнями поточкових джерел одночасно, а також виконує обробку сигналу з високою роздільною здатністю без затримок. Додаткові функції програмного

забезпечення vMix, як-от хромакей, віртуальні студії, ефекти уповільнення та повтору, значно розширюють можливості обробки відеопотоку на етапі трансляції [7]-[10], [13].

Серед інших особливостей варто відзначити підтримку мережевого обміну відео, вбудовані інструменти композитингу, мікшування звуку, генерації титрів і анімації. Високий рівень інтеграції з Інтернет протоколами потокової передачі дозволяє транслювати відеопотік на різні платформи без потреби у проміжних етапах збереження або повторного завантаження. Усе це створює умови для повноцінної обробки відеопотоку від джерела до кінцевого споживача [7]-[10], [13].

Завдяки інтуїтивно зрозумілому інтерфейсу програмне забезпечення забезпечує контроль над усіма елементами потокової трансляції — від вибору джерел і компоновки сцени до мікшування звуку і моніторингу якості потоку. Незважаючи на те, що середовищем роботи є виключно операційна система Windows, можливість інсталяції через додаткові інструменти дозволяє розширити сферу застосування. У сукупності всі можливості vMix роблять його ефективним рішенням для повноцінної обробки відеопотоків у професійному середовищі [7]-[10], [13].

Основними характеристиками програмного забезпечення vMix є такі [7]-[10], [13]:

- підтримка високої якості відео: програмне забезпечення vMix підтримує обробку відеопотоків з роздільною здатністю до 4K, що дозволяє транслювати чітке та деталізоване зображення [7]-[10], [13];

- сумісність із різними джерелами відео: програмне забезпечення vMix дозволяє використовувати вебкамери, зовнішні камери, пристрої захоплення, списки відтворення, презентації, фотографії та DVD [7]-[10], [13];

- потужні засоби відеообробки: програмне забезпечення vMix включає функції віртуальних студій, ефектів сповільненої зйомки та миттєвого повтору [7]-[10], [13];

- інтеграція з потоковими платформами: програмне забезпечення vMix підтримує трансляцію на популярні сервіси, такі як YouTube, Facebook, Twitch за допомогою протоколів RTMP та HLS [7]-[10], [13];

- мережеві можливості: програмне забезпечення vMix забезпечує передавання відеопотоків у локальній мережі з низькою затримкою та високою якістю [7]-[10], [13];

- вбудований аудіомікшер: програмне забезпечення vMix дозволяє керувати аудіосигналами з різних джерел у режимі реального часу [7]-[10], [13];

- інструменти створення графіки: програмне забезпечення vMix містить функції для генерації титрів, анімації через XAML та оформлення сцен [7]-[10], [13];

- інтеграція із соціальними мережами: програмне забезпечення vMix дає змогу виводити коментарі та повідомлення з соціальних платформ у відеопотік за допомогою vMix Social [7]-[10], [13];

- адаптація до різних потреб: програмне забезпечення vMix доступне у кількох варіантах ліцензії, що дозволяє вибрати оптимальний набір функцій для конкретного проєкту [7]-[10], [13];

- можливість запису трансляцій: програмне забезпечення vMix підтримує збереження відеопотоків на диск для подальшого використання або редагування [7]-[10], [13];

- інтерфейс користувача: програмне забезпечення vMix забезпечує зручне керування сценами, переходами та джерелами відео навіть під час прямого ефіру [7]-[10], [13];

- сумісність із Windows: програмне забезпечення vMix працює виключно в середовищі цієї операційної системи, але може бути встановлене на Mac за допомогою BootCamp [7]-[10], [13].

Серед недоліків програмного забезпечення vMix можна відзначити досить високу вимогливість до апаратного забезпечення, що може створювати складнощі при роботі на менш потужних комп'ютерах. Крім того, повноцінне

опанування всіх функцій потребує певного часу та технічної підготовки, особливо для користувачів без досвіду в обробці відеопотоків. Обмеження підтримки лише операційною системою Windows також звужує коло потенційних користувачів, оскільки програмне забезпечення не має нативної версії для macOS чи Linux [7]-[10], [13].

Програмне забезпечення D3DGear (рис. 1.2) призначене для ефективної обробки відеопотоків, зокрема в контексті запису й трансляції ігрового процесу в реальному часі. Його функціонування оптимізоване таким чином, щоб не створювати помітного навантаження на систему, що дозволяє забезпечувати плавну трансляцію відеопотоків без втрати якості зображення чи зниження частоти кадрів. Завдяки вбудованим механізмам апаратного прискорення, забезпечується надійна передача відеопотоку навіть за умов обмеженої швидкості Інтернет з'єднання [9], [12], [14].

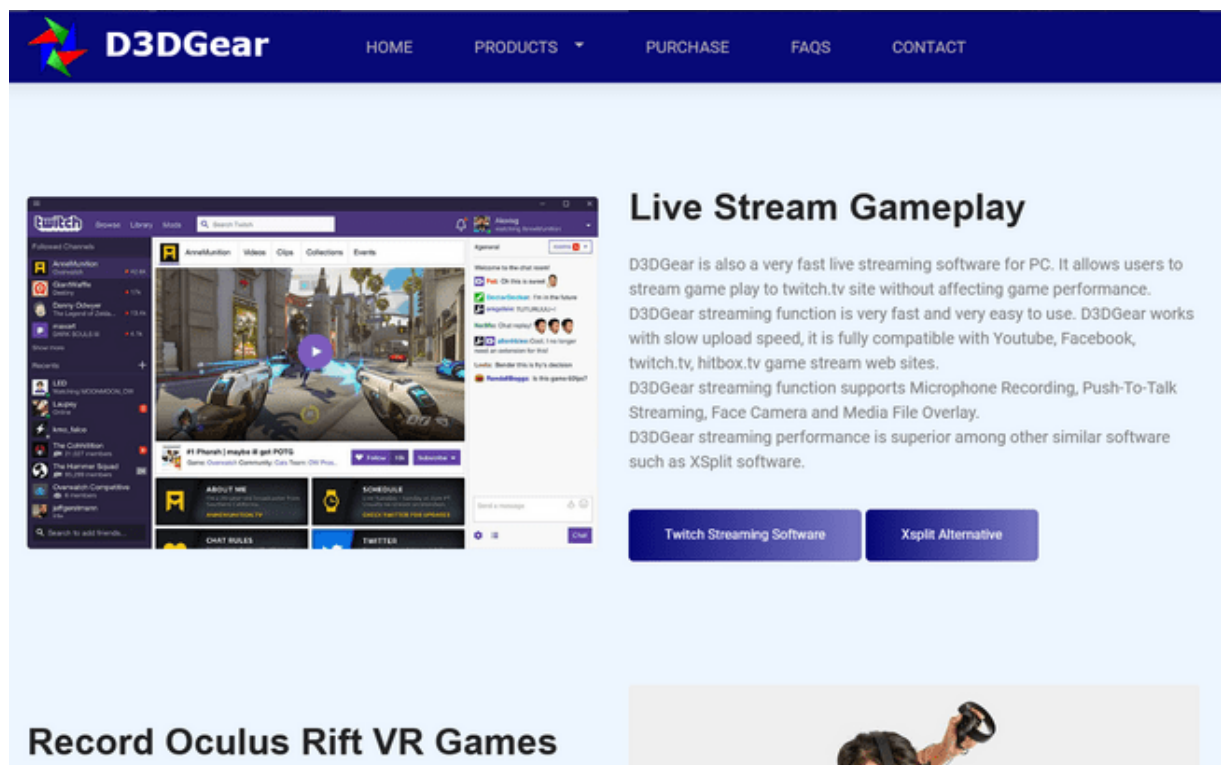


Рисунок 1.2 – Програмне забезпечення D3DGear [14]

Програмне забезпечення D3DGear взаємодіє з популярними платформами для потокової передачі контенту, що дає змогу безпосередньо

передавати відеопотік на сайти, орієнтовані на трансляцію ігор. Підтримується широкий спектр графічних технологій і API, що забезпечує стабільну обробку відеопотоків незалежно від використовуваного графічного рушія. Інтерфейс налаштувань дозволяє змінювати ключові параметри відеопотоку, зокрема роздільну здатність, якість стиснення та частоту кадрів [9], [12], [14].

Для обробки відеопотоків застосовуються як системні кодеки, так і апаратні енкодери провідних виробників, що забезпечує високу ефективність і якість відеопередачі. Після завершення трансляції оброблений відеопотік може бути збережений у зручному форматі для подальшого використання. Програма також підтримує захоплення аудіопотоків, що дозволяє здійснювати синхронну обробку відео та звуку під час прямої трансляції [9], [12], [14].

Програмне забезпечення D3DGeaг має такі особливості [9], [12], [14]:

- низьке навантаження на систему: програмне забезпечення D3DGeaг забезпечує мінімальний вплив на продуктивність комп'ютера під час обробки відеопотоків [9], [12], [14];
- висока якість запису: програмне забезпечення D3DGeaг підтримує збереження відеопотоків у високій роздільній здатності без втрати кадрів чи затримки [9], [12], [14];
- підтримка апаратного кодування: програмне забезпечення D3DGeaг підтримує використання графічних процесорів AMD, NVIDIA та Intel для обробки відеопотоків у режимі реального часу [9], [12], [14];
- сумісність із поточковими платформами: програмне забезпечення D3DGeaг забезпечує передачу відеопотоків безпосередньо на популярні сервіси поточної трансляції [9], [12], [14];
- налаштування відео та аудіо: програмне забезпечення D3DGeaг надає можливість регулювання параметрів відеопотоку, таких як частота кадрів, роздільна здатність та формат звуку [9], [12], [14];
- підтримка ігрових API: програмне забезпечення D3DGeaг підтримує обробку відеопотоків із застосуванням DirectX, OpenGL, Mantle та інших графічних інтерфейсів [9], [12], [14];

- збереження відеофайлів у різних форматах: програмне забезпечення D3DGeaг підтримує формати AVI, AVI2 та WMV для локального збереження відеопотоків після трансляції [9], [12], [14];
- запис ігрового аудіо: програмне забезпечення D3DGeaг забезпечує синхронну обробку звукового потоку разом із відео потоком під час запису ігор [9], [12], [14];
- підтримка роботи з повільним Інтернетом: оптимізована передача відеопотоків навіть за умов низької швидкості з'єднання [9], [12], [14];
- зручність використання: інтерфейс програмного забезпечення D3DGeaг дозволяє швидко розпочинати запис та трансляцію без складних налаштувань [9], [12], [14].

Серед недоліків програмного забезпечення D3DGeaг можна відзначити обмежені можливості редагування відеопотоків після запису, відсутність сучасного інтерфейсу користувача, а також обмежену інтеграцію з деякими новими платформами потокової передачі. Крім того, деякі користувачі можуть відчувати нестачу додаткових функцій для професійної обробки відео або стримінгу, які доступні в більш комплексних рішеннях [9], [12], [14].

Програмне забезпечення Ecamт (рис. 1.3) створено для обробки відеопотоків і забезпечує зручне середовище для створення, редагування та трансляції мультимедійного контенту. Програмне забезпечення Ecamт орієнтоване на ефективну роботу з поточним відео у реальному часі, надаючи інструменти для запису, трансляції, виведення аудіо- та відеосигналів, а також візуального оформлення трансляцій. Завдяки інтеграції з популярними платформами для потокової передачі відео, забезпечується легкий доступ до створення живих ефірів без зайвих технічних ускладнень. Через можливість обробки відеопотоків з декількох камер, додаванням анімацій, ефектів, таймерів, графічних накладок та елементів управління, програмне забезпечення Ecamт активно застосовують для проведення інтерв'ю, вебінарів, презентацій, онлайн заходів і подкастів [7], [8], [10], [15].



Рисунок 1.3 – Програмне забезпечення Ecamm [15]

Під час обробки відеопотоків забезпечується підтримка зовнішніх пристроїв, таких як DSLR-камери, пристрої HDMI-захоплення, мікрофони, а також мобільні пристрої як джерела сигналу. Також передбачено можливість змішування відео- та аудіопотоків, що дозволяє створювати якісний ефір у реальному часі. Функціонал включає в себе елементи керування роздільною здатністю, форматом потоку та параметрами трансляції, що дозволяє налаштувати оптимальну якість зображення і звуку відповідно до обмежень платформи чи обладнання [7], [8], [10], [15].

Програмне забезпечення Ecamm активно використовується для потокової обробки відео завдяки простоті керування, можливості підключення гостей, додавання інтерактивних елементів у прямий ефір і збереження відео для подальшого використання або монтажу. Програмне забезпечення Ecamm також дозволяє зручно керувати розкладом трансляцій, забезпечуючи професійний рівень представлення контенту в реальному часі. Таким чином, програмне забезпечення Ecamm виступає як потужний інструмент для повноцінної обробки відеопотоків у процесі запису, трансляції та постпродакшну [7], [8], [10], [15].

Програмне забезпечення Ecamm має такі особливості [7], [8], [10], [15].

- підтримка потокового відео: програмне забезпечення Ecamm забезпечує трансляцію на платформи YouTube, Facebook, Twitch, LinkedIn, X (Twitter) та інші [7], [8], [10], [15];

- обробка відеопотоків у реальному часі: програмне забезпечення Ecamm дозволяє змінювати сцени, додавати ефекти, анімації, коментарі та відео накладки без переривання трансляції [7], [8], [10], [15];

- інтеграція з зовнішніми пристроями: програмне забезпечення Ecamm підтримує підключення DSLR-камер, мікрофонів, пристроїв HDMI-захоплення, iOS-пристроїв, Elgato Stream Deck [7], [8], [10], [15];

- підтримка багатокамерної трансляції: програмне забезпечення Ecamm дозволяє перемикатися між різними джерелами відеопотоків під час ефіру у режимі реального часу [7], [8], [10], [15];

- сумісність із Skype: програмне забезпечення Ecamm надає можливість запрошення до п'яти гостей в інтерв'ю або панельні сесії у режимі реального часу [7], [8], [10], [15];

- функція віртуальної камери: програмне забезпечення Ecamm дозволяє використовувати Ecamm Live як джерело відео в сторонніх застосунках типу Zoom чи Microsoft Teams (у плані Pro) [7], [8], [10], [15];

- підтримка запису: програмне забезпечення Ecamm здійснює локальний запис відео та аудіо, зберігаючи високоякісні копії онлайн трансляцій [7], [8], [10], [15];

- налаштування трансляції: програмне забезпечення Ecamm надає можливості для налаштування фонів, шарів, макетів, а також використання ефектів і таймерів [7], [8], [10], [15];

- підтримка формату RTMP: програмне забезпечення Ecamm дозволяє транслювати на нестандартні платформи через ручне налаштування відео потоку [7], [8], [10], [15];

– користувацький інтерфейс: програмне забезпечення Ecamm має просте й інтуїтивно зрозуміле управління, що підходить як початківцям, так і професіоналам [7], [8], [10], [15].

Серед недоліків програмного забезпечення Ecamm можна відзначити відсутність мобільного застосунку для iPhone, що обмежує можливості мобільної роботи з відеопотоками. Деякі функції, такі як віртуальна камера або моніторинг у реальному часі, доступні лише у дорожчому тарифному плані, що може бути незручним для користувачів з базовими потребами. Крім того, процес налаштування шарів, макетів і фонів може здатися неінтуїтивним, особливо для новачків, а експорт конфігурацій для використання на інших пристроях не завжди працює коректно [7], [8], [10], [15].

Порівняльну характеристику програмного забезпечення для обробки відеопотоків наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння програмного забезпечення для обробки відеопотоків

Критерій порівняння	vMix [13]	D3DGear [14]	Ecamm [15]
Підтримка декількох камер	+	—	+
Підтримка ігрових відеопотоків	+—	+	—
Апаратне кодування	+	+	+—
Підтримка високої роздільної здатності	+	+—	+
Зручність інтерфейсу	+—	+	+
Підтримка мобільних пристроїв	+—	—	—

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато програмних засобів для обробки відеопотоків. Проте деякі програмні засоби для обробки відеопотоків мають низку недоліків, які слід враховувати під час впровадження та використання. Однією

з головних проблем є висока вимогливість до апаратного забезпечення. Обробка відео, особливо в реальному часі, потребує значних обчислювальних ресурсів, зокрема продуктивних процесорів, графічних прискорювачів і великої кількості оперативної пам'яті. Це підвищує загальну вартість системи й ускладнює її масштабування, особливо для малих підприємств або у випадках обробки відео на периферійних пристроях із обмеженими ресурсами. Ще одним суттєвим недоліком є складність у налаштуванні та інтеграції деяких програмних засобів для обробки відеопотоків. Багато програмних рішень потребують технічних знань для інсталяції, конфігурації та обслуговування. У деяких випадках потрібна участь спеціалістів із машинного навчання, комп'ютерного зору або мережевої безпеки, що обмежує доступність цих технологій для непрофесійних користувачів або невеликих команд. Тому актуальною є розробка програмного забезпечення для обробки відеопотоків.

При розробці програмного забезпечення для обробки відеопотоків необхідно забезпечити такі функціональні вимоги:

- підтримка можливості створення нової відеотрансляції;
- підтримка можливості авторизації та реєстрації користувачів;
- можливість передачі відеопотоків;
- можливість вибору параметрів відеотрансляції (зокрема, пристроїв з яких забезпечується передача відеопотоків);
- підтримка можливості налаштування інтерфейсу у процесі відеотрансляції;
- підтримка можливості запису відеопотоку у файл для подальшого відтворення.

1.3 Висновки за розділом 1

Визначено, що відеопотік передає безпосередньо через мережу фрагменти мультимедійних даних, що дає змогу почати перегляд відразу ж

після ініціалізації відтворення, без потреби чекати на завантаження цілого файлу. Програмне забезпечення для обробки відеопотоків являє собою сукупність програмних рішень, які дозволяють приймати, аналізувати, редагувати, трансформувати або передавати відео в режимі реального часу або з мінімальною затримкою. Основна мета такого ПЗ полягає у забезпеченні безперервної роботи з відеопотоком, тобто потоком кадрів, що надходять у реальному часі, зберігаючи його якість та синхронізацію.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для обробки відеопотоків. Проте деякі програмні засоби для обробки відеопотоків мають низку недоліків, які слід враховувати під час впровадження та використання. Однією з головних проблем є висока вимогливість до апаратного забезпечення. Обробка відео, особливо в реальному часі, потребує значних обчислювальних ресурсів, зокрема продуктивних процесорів, графічних прискорювачів і великої кількості оперативної пам'яті. Це підвищує загальну вартість системи й ускладнює її масштабування, особливо для малих підприємств або у випадках обробки відео на периферійних пристроях із обмеженими ресурсами. Ще одним суттєвим недоліком є складність у налаштуванні та інтеграції деяких програмних засобів для обробки відеопотоків. Багато програмних рішень потребують технічних знань для інсталяції, конфігурації та обслуговування. У деяких випадках потрібна участь спеціалістів із машинного навчання, комп'ютерного зору або мережевої безпеки, що обмежує доступність цих технологій для непрофесійних користувачів або невеликих команд. Тому актуальною є розробка програмного забезпечення для обробки відеопотоків.

Сформульовано функціональні вимоги до програмного забезпечення для обробки відеопотоків.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмного забезпечення для обробки відеопотоків було використано мову програмування JavaScript [16], [17].

JavaScript – вважається однією з найвідоміших мов програмування сучасності, яка за роки свого розвитку набула широкого застосування в різних сферах інформаційних технологій. Хоча початково її функціональність обмежувалась виключно створенням інтерактивних елементів на вебсторінках, з часом було доведено, що ця мова здатна вирішувати значно складніші завдання. Сьогодні її використовують не тільки у веброзробці, а й у побудові інтерфейсів для десктопних застосунків, в ігровій індустрії, у створенні програмного забезпечення для Інтернету речей, а також у розробці серверних рішень [16], [17].

Завдяки вбудованій підтримці в усіх сучасних вебпереглядачах забезпечується миттєвий запуск JavaScript-коду без додаткових інструментів, що сприяє зручності та швидкодії вебпрограм. З появою Node.js стало можливим перенести виконання цієї мови за межі браузера, що дало змогу писати повноцінні серверні застосунки тими самими засобами, що й фронтенд. Це суттєво зменшило складність розробки й спростило підтримку кодової бази [16], [17].

Екосистема мови JavaScript охоплює велику кількість готових рішень, серед яких варто згадати такі популярні інструменти, як React, Angular і Vue.js. Їх використання дозволяє не тільки прискорити процес створення інтерфейсів, а й забезпечити масштабованість, повторне використання компонентів та адаптацію до вимог великомасштабних проєктів. Жвава спільнота підтримує розвиток мови, пропонує численні навчальні ресурси, а також забезпечує постійні оновлення, що вдосконалюють синтаксис і функціональні можливості [16], [17].

Мова програмування JavaScript підтримує різні парадигми програмування, зокрема імперативну, функціональну та об'єктно-орієнтовану, що робить її придатною до використання у проєктах із різною структурою. Завдяки можливості роботи з асинхронним кодом, включно з механізмами обробки обіцянок, асинхронних функцій і викликів, забезпечується ефективна взаємодія з API та потоками даних у реальному часі [16], [17].

Водночас велика свобода, яку надає мова програмування JavaScript, може вимагати додаткових зусиль для підтримання чистоти та якості коду. З цією метою до процесу розробки залучаються зовнішні інструменти, серед яких системи тестування, статичний аналіз, компілятори та автоматизовані скрипти [16], [17].

У підсумку, мова програмування JavaScript постає як багатофункціональний програмний інструмент, що поєднує в собі зручність, гнучкість і підтримку численних сценаріїв використання, завдяки чому утримує провідні позиції серед мов програмування для створення сучасного програмного забезпечення [16], [17].

Використання мови програмування JavaScript у проєктах, пов'язаних із потоковим відео, розглядається як доцільне рішення завдяки технічній відповідності вимогам вебсередовища та широким функціональним можливостям. Оскільки ця мова є нативною для браузера, її застосування дозволяє організовувати відеопередачу за допомогою таких технологій, як WebRTC, WebGL і WebAssembly, що підтримуються сучасними стандартами HTML5. Завдяки цьому забезпечується стабільна робота вебзастосунків на різних пристроях без необхідності встановлення додаткового програмного забезпечення, що дозволяє охопити широку аудиторію та забезпечити доступність сервісу [16], [17].

Під час роботи з відеоданими в браузері застосовується клієнтська обробка, яка не тільки дозволяє відтворювати зображення та звук у режимі реального часу, а й дає змогу реалізовувати попередній аналіз або обробку кадрів без суттєвого навантаження на сервер. Це особливо корисно в

сценаріях, де потрібно знизити затримку або уникнути втрати інформації при передачі. Крім того, використання бібліотек машинного навчання, таких як TensorFlow.js, дає змогу безпосередньо в браузері виконувати складні обчислення, пов'язані з виявленням об'єктів, розпізнаванням дій чи класифікацією відеоконтенту [16], [17].

Окрім технічних переваг, слід відзначити активну підтримку з боку спільноти JavaScript-розробників, яка постійно створює нові інструменти та компоненти, що значно полегшують інтеграцію відеопотоків з іншими вебсервісами. Завдяки готовим рішенням і гнучким API відкривається можливість розробки масштабованих вебдодатків, які можуть ефективно працювати у хмарних середовищах та підтримувати інтеграцію з зовнішніми платформами [16], [17].

Обґрунтування вибору мови програмування для розробки програмного забезпечення для обробки відеопотоків наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки програмного забезпечення для обробки відеопотоків

Критерій порівняння мов програмування	Мова програмування		
	JavaScript	Python	C++
Підтримка обробки відео в браузері	+	+—	—
Кросплатформеність клієнтських застосунків	+	+—	+—
Доступність бібліотек для відеоаналітики	+	+	+—
Зручність для створення ПЗ для обробки відеопотоків	+	—	—
Адаптація до хмарних середовищ	+	+	+—

Отже, для реалізації програмного забезпечення для обробки відеопотоків обрано мову програмування JavaScript. Вибір мови JavaScript у ролі основного інструменту для реалізації функціоналу обробки потокового відео ґрунтується на поєднанні високої сумісності з вебтехнологіями, можливості виконання складних обчислень без виходу за межі браузера, широкого інструментарію та активного розвитку екосистеми, що робить його вдалим вибором у цій галузі.

2.2 Вибір бібліотеки для створення програмного забезпечення для обробки відеопотоків

Для створення програмного забезпечення для обробки відеопотоків було обрано бібліотеку React [18], [19].

Бібліотека React є не лише інструментом для створення користувацького інтерфейсу, а і повноцінною платформою з розвиненим середовищем, що відкриває широкі можливості для побудови гнучких і масштабованих вебрішень. У контексті взаємодії з даними зміни стану обробляються автоматично, що дозволяє уникнути ручного оновлення елементів сторінки — усі модифікації відображаються у візуальному представленні за допомогою механізму віртуального об'єктного механізму, який оптимізує взаємодію з браузером, зменшуючи кількість звернень до реального інтерфейсу [18], [19].

Цей підхід дозволяє досягати високої продуктивності, навіть коли йдеться про складні інтерфейси з численними динамічними компонентами. Концепція реактивності вбудована на рівні логіки: компоненти автоматично реагують на зміни стану, що відкриває можливості для реалізації систем, здатних адаптуватися до динамічних даних. Контроль стану здійснюється централізовано, що полегшує передбачуване оновлення інтерфейсу [18], [19].

Бібліотека React не нав'язує жорстких структур даних під час побудови застосунків, завдяки чому рішення можуть проєктуватись у довільний спосіб,

відповідно до технічних або бізнес-вимог. При цьому у розробника залишається можливість вільно поєднувати його з іншими бібліотеками чи фреймворками, що дозволяє використовувати найбільш ефективні засоби для конкретного випадку [18], [19].

Опанування програмної бібліотеки React не потребує значного часу: синтаксис JSX дає змогу легко зрозуміти логіку побудови інтерфейсів, оскільки елементи описуються в структурі, подібній до HTML, але на основі JavaScript. Завдяки численним прикладам, відкритій документації та активній спільноті, вивчення React стає доступним як для новачків, так і для досвідчених розробників [18], [19].

Сфера застосування не обмежується лише браузером. Існує можливість створення мобільних застосунків із використанням React Native, що дає змогу охопити користувачів різних платформ — від смартфонів до настільних пристроїв. У результаті, забезпечується плавна та уніфікована взаємодія незалежно від середовища, у якому працює застосунок [18], [19].

Бібліотека React розглядається як сучасне середовище для побудови вебінтерфейсів, яке використовує JavaScript і орієнтоване на створення динамічних додатків з інтерактивною структурою. Його розроблено компанією Meta з метою забезпечення ефективної взаємодії користувача з динамічним контентом. Архітектура React ґрунтується на концепції компонентів — незалежних елементів з власною логікою та станом, які легко комбінуються й масштабуються, що забезпечує гнучкість у розробці [18], [19].

У контексті застосувань, пов'язаних із обробкою відео та візуалізацією складних даних, цей підхід надає ефективні засоби для керування змінами інтерфейсу в реальному часі. Компонентна модель дозволяє ізольовано контролювати окремі частини сторінки, що особливо важливо при роботі з чутливими даними, такими як візуальний контент, часові мітки, текстова аналітика або графічні анотації. Вбудовані механізми передачі даних між компонентами спрощують підтримку логіки взаємодії та роблять оновлення стану передбачуваними [18], [19].

Можливості бібліотеки React значно розширюються завдяки великій кількості зовнішніх бібліотек і розширень, які активно застосовуються для інтеграції з WebRTC, організації потокового зв'язку, впровадження елементів штучного інтелекту безпосередньо в браузері або обробки відео на клієнтській стороні [18], [19].

Таким чином, завдяки поєднанню високої продуктивності, гнучкої архітектури й підтримки потужної екосистеми інструментів, React залишається одним із провідних рішень для реалізації додатків, що працюють із потоковим відео та вимагають високого ступеня інтерактивності [18], [19].

Обґрунтування вибору бібліотеки для створення програмного забезпечення для обробки відеопотоків наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору бібліотеки для створення програмного забезпечення для обробки відеопотоків

Критерій порівняння бібліотек	Бібліотеки		
	React	Vue.js	Angular
Гнучкість архітектури та налаштування	+	+	+—
Продуктивність в динамічних інтерфейсах	+	+—	+—
Ефективна робота з потоковим відео в реальному часі	+	+—	+—
Наявність компонентів для роботи з відео	+	+	+—
Сумісність з мобільною розробкою	+	—	+—
Простота вивчення і використання	+	+	+—

Таким чином, для створення програмного забезпечення для обробки відеопотоків обрано бібліотеку React через високу продуктивність і здатність ефективно оновлювати змінені частини інтерфейсу без перезавантаження всієї сторінки. Це особливо важливо, коли йдеться про роботу з динамічними даними, як-от відеопотоки, де постійно змінюються кадри, таймлайни, елементи керування відтворенням, аналітика чи анотації. Крім того, React дозволяє зручно керувати станом застосунку і швидко передавати дані між компонентами, що дає змогу реалізовувати складну логіку взаємодії з відео без втрати швидкодії та зручності користувача.

2.3 Висновки за розділом 2

Для реалізації програмного забезпечення для обробки відеопотоків обрано мову програмування JavaScript. Вибір мови JavaScript у ролі основного інструменту для реалізації функціоналу обробки потокового відео ґрунтується на поєднанні високої сумісності з вебтехнологіями, можливості виконання складних обчислень без виходу за межі браузера, широкого інструментарію та активного розвитку екосистеми, що робить його вдалим вибором у цій галузі.

Для створення програмного забезпечення для обробки відеопотоків обрано бібліотеку React через високу продуктивність і здатність ефективно оновлювати змінені частини інтерфейсу без перезавантаження всієї сторінки. Це особливо важливо, коли йдеться про роботу з динамічними даними, як-от відеопотоки, де постійно змінюються кадри, таймлайни, елементи керування відтворенням, аналітика чи анотації. Крім того, React дозволяє зручно керувати станом застосунку і швидко передавати дані між компонентами, що дає змогу реалізовувати складну логіку взаємодії з відео без втрати швидкодії та зручності користувача.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення для обробки відеопотоків

Структуру програмного забезпечення для обробки відеопотоків наведено на рис. 3.1.

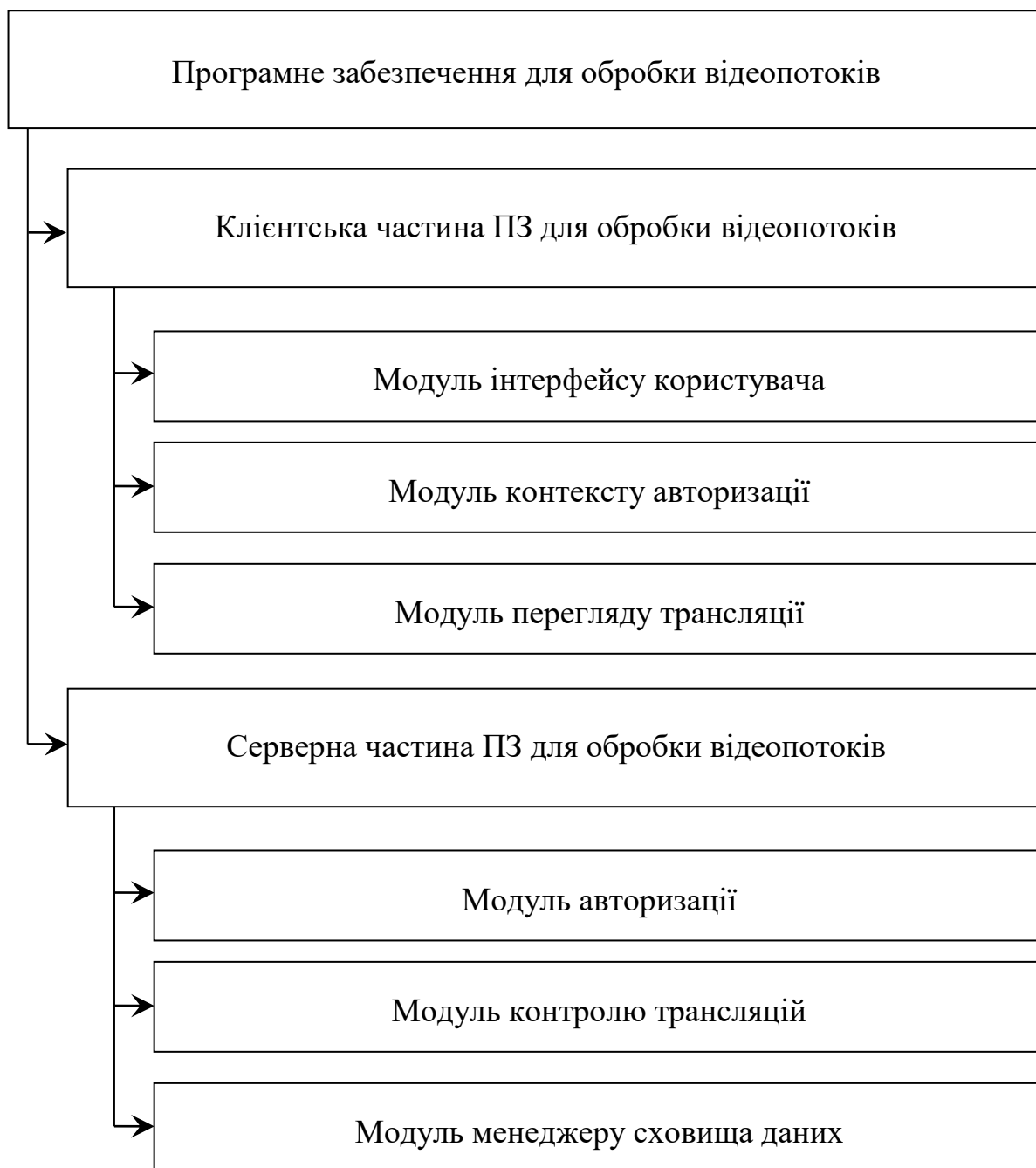


Рисунок 3.1 – Структура програмного забезпечення для обробки відеопотоків

Програмне забезпечення для обробки відеопотоків реалізується як багаторівнева система, де взаємодія між клієнтом і сервером забезпечує цілісну логіку роботи додатку. У межах клієнтської частини структура коду безпосередньо відображає функціональний поділ інтерфейсу: спільні компоненти, як-от головне меню чи панель навігації, є універсальними для всіх розділів, тоді як окремі сторінки — авторизації, трансляції, створення подій — мають вузьке прив'язування до відповідних функціональних модулів. Завантаження певних сегментів виконується динамічно лише за наявності автентифікації, що дає змогу зменшити початкове навантаження на клієнт.

У структурі клієнта ключову роль відіграє стартова логіка, яка ініціює завантаження ресурсів, підключає базові сервіси, обробляє глобальні параметри й виконує початкове відображення. Подальше переміщення між розділами здійснюється через централізований навігаційний механізм, який визначає, які саме компоненти та модулі мають бути активовані залежно від маршруту та контексту. Завдяки динамічному імпорту, окремі частини застосунку не завантажуються, поки не виникне потреба в їх використанні. У середині кожного функціонального блоку розміщено ізольовану логіку, що дозволяє уникнути зайвих залежностей і полегшує підтримку коду.

Дані, які передаються між модулями, зберігаються за допомогою спеціалізованих об'єктів, що відповідають за управління станом і автоматичну реакцію на його зміни. Ці контекстні об'єкти формуються з урахуванням призначення конкретної функції — вони обслуговують як навігаційні механізми, так і теми, налаштування інтерфейсу або обробку відео. Їхнє використання дає змогу організувати реактивний рендеринг та забезпечити цілісність поведінки додатку.

На серверному рівні архітектура не зберігає стану користувача між запитами, що дозволяє масштабувати обробку паралельно. Усі запити сприймаються як незалежні події, які перевіряються за параметрами, методом звернення та авторизаційною інформацією. Обробка вхідних звернень

виконується відповідно до логіки конкретного модуля, без централізованого контролю.

Структура серверної частини вибудовується за принципом функціонального розмежування відповідальності. Кожен модуль виконує власну задачу, але в окремих випадках залежить від результатів інших компонентів, як-от перевірка автентичності користувача перед доступом до закритих ресурсів. Робота сервера ґрунтується на модульності — обробка потоків, обробка користувачів та зберігання даних реалізовані у вигляді окремих сегментів. Запуск і конфігурування застосунку виконується через відповідний фреймворк, який забезпечує ініціалізацію сервісів і контроль залежностей.

Механізм автентифікації опирається на сучасні протоколи й керує всім, що пов'язано з управлінням користувачем, — від реєстрації до верифікації доступу. Модуль збереження надає інтерфейси для роботи з файлами, спочатку використовуючи локальне зберігання, з можливістю подальшого масштабування на зовнішні сервіси. Найбільш комплексний сегмент системи — це блок, що відповідає за трансляцію: він включає логіку підключення, обмін повідомленнями через веб-сокети, підтримку зв'язку з сервером кодування та обробку подій, пов'язаних із керуванням відеоконтентом у реальному часі.

Таким чином, обробка відеопотоків організовується через взаємодію кількох автономних модулів, кожен з яких виконує строго окреслену функцію в загальній архітектурі системи.

3.2 Опис модулів програмного забезпечення для обробки відеопотоків

У клієнтській частині програмного забезпечення, яке обробляє відеопотоки, основна увага приділяється взаємодії користувача з інтерфейсом,

завантаженню динамічних компонентів та реактивному оновленню контенту. До ключових модулів належать такі.

Модуль інтерфейсу користувача відповідає за побудову зовнішнього вигляду застосунку: компоненти навігації, вікна трансляцій, сторінки профілю та системні повідомлення. Він використовує сучасні фреймворки (наприклад, React або Vue) та забезпечує реактивне оновлення сторінки на основі змін стану. Особливістю є компонентна архітектура, де навіть найменші частини інтерфейсу (кнопки, модальні вікна) реалізовані як самостійні блоки.

Модуль контексту авторизації призначений для зберігання й обробки стану автентифікації користувача. Цей модуль ініціалізується під час завантаження застосунку, перевіряє токен у локальному сховищі та за його наявності підтверджує авторизацію. Контекст дозволяє забезпечити доступ до захищених сторінок лише після підтвердження особи. Усі запити до серверу автоматично включають авторизаційний токен.

Модуль перегляду трансляцій забезпечує відображення відеопотоків у реальному часі, управління якістю зображення. Також підтримує інтерактивні елементи: чат, лайки, реакції. Особливістю є адаптивність до пропускну здатності каналу й автоматичне перемикання між джерелами відео в разі зміни мережевого середовища.

Серверна частина програми забезпечує логіку доступу, обробку запитів, зберігання медіа та координацію потокових підключень. Серед її модулів вирізняються такі.

Модуль авторизації реалізує механізми реєстрації, входу в систему, валідації токенів та керування сесіями. Побудований на базі стандартів OAuth2 або JWT. Забезпечує захист приватних API-ендпоінтів та дає змогу гнучко керувати рівнем доступу користувачів до функцій застосунку. Має інтеграцію з базою даних облікових записів та логуванням спроб входу.

Модуль контролеру трансляцій обробляє запити, пов'язані з запуском, зупинкою та моніторингом відеопотоків. Він комунікує з медіасервером, генерує URL-адреси потоків і веде облік активних трансляцій. Особливість —

підтримка масштабування на кілька серверів через балансувальник навантаження та логіка контролю якості трансляцій.

Модуль менеджера сховища призначений для збереження записаних відео, скріншотів трансляцій, метаданих подій. Підтримує зберігання як на локальних носіях, так і на хмарних платформах. Додатково реалізує механізм очищення застарілих даних та резервного копіювання. Має інтеграцію з CDN для прискорення доставки контенту користувачам.

Загальна особливість архітектури полягає в чіткому поділі ролей: клієнт оперує станом інтерфейсу та подає запити, тоді як сервер обробляє логіку, управляє безпекою, потоками і збереженням відеоданих. Така структура дозволяє масштабувати систему, розгортати її у хмарній інфраструктурі та забезпечувати високий рівень стабільності.

Отже програмне забезпечення для обробки відеопотоків реалізується як багаторівнева система, де взаємодія між клієнтом і сервером забезпечує цілісну логіку роботи програми. У клієнтській частині програмного забезпечення, яке обробляє відеопотоки, основна увага приділяється взаємодії користувача з інтерфейсом, завантаженню динамічних компонентів та реактивному оновленню контенту. До ключових модулів належать: модуль інтерфейсу користувача, модуль контексту авторизації, модуль перегляду трансляцій. Серверна частина програми забезпечує логіку доступу, обробку запитів, зберігання медіа та координацію поточкових підключень. Серед її модулів вирізняються такі: модуль авторизації, модуль контролера трансляцій, модуль менеджера сховища.

3.3 Функціонування програмного забезпечення для обробки відеопотоків

Функціонування програмного забезпечення для обробки відеопотоків подамо за допомогою схеми, зображеної на рис. 3.2.



Рисунок 3.2 – Схема функціонування програмного забезпечення для обробки відеопотоків

Робота програмного забезпечення для обробки відеопотоків починається з відкриття користувачем адреси вебзастосунку. У середовищі розробки це зазвичай локальний хост, що вказує на порт, зарезервований для клієнтської частини. Всі взаємодії з програмою зводяться до переходів між окремими сторінками інтерфейсу через панель навігації або керувальні елементи, що відповідають за зміну стану застосунку.

На першому етапі взаємодії обов'язковим є процес автентифікації: користувач проходить процедуру входу або створює новий обліковий запис. Усі дії перевіряються системою на відповідність правилам — наявність усіх

обов'язкових полів, коректність даних та унікальність облікового запису. Доступ до внутрішніх сторінок застосунку відкривається виключно після успішної автентифікації, тоді як доступ до форм авторизації блокується для вже підключених користувачів.

Після входу у систему відбувається перенаправлення до головного розділу, звідки стає доступною функція створення трансляційної події. Після виклику відповідного інтерфейсу відображається форма, де користувач має змогу задати назву події та, за бажанням, вказати короткий опис. У разі відмови, передбачено можливість повернення на попередній етап без збереження змін. Система обмежує користувача в кількості активних подій, тому нова може бути створена лише за умови відсутності незавершеної попередньої. Якщо подія вже існує, здійснюється автоматичне перенаправлення до відповідного інтерфейсу її редагування.

Далі запускається інтерфейс управління подією, який дозволяє обирати джерела аудіо та відео, переглядати поточне зображення з камер, змінювати або вимикати пристрої захоплення у будь-який момент. Передбачено можливість взаємодії з графічними елементами на екрані – їх можна додавати, редагувати, переміщувати та вилучати. За потреби можна формувати і зберігати окремі набори графічних об'єктів з подальшим швидким увімкненням. Паралельно забезпечено контроль глобальних параметрів трансляції та перегляд стану відеопотоку й надійності підключення до сервера. Для зручності передбачено режим, в якому елементи керування ховаються, щоб збільшити площу зображення для моніторингу.

Фінальним етапом є запуск трансляції, під час якого інтерфейс зберігає всі основні функції налаштування. Завершення сесії супроводжується переходом до сторінки з повідомленням про завершення події. У поточній версії програмне забезпечення обмежується функціями автентифікації, ініціалізації трансляції, управління графічним вмістом та базовим контролем стану відеопотоку.

3.4 Проєктування інтерфейсу програмного забезпечення для обробки відеопотоків

Проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для обробки відеопотоків виконано з урахуванням вимог технічного завдання. При розробленні програмного забезпечення для обробки відеопотоків враховані функціональні вимоги до програми:

- підтримка можливості створення нової відеотрансляції;
- підтримка можливості авторизації та реєстрації користувачів;
- можливість передачі відеопотоків;
- можливість вибору параметрів відеотрансляції (зокрема, пристроїв з яких забезпечується передача відеопотоків);
- підтримка можливості налаштування інтерфейсу у процесі відеотрансляції;
- підтримка можливості запису відеопотоку у файл для подальшого відтворення.

Інтерфейс головної сторінки програмного забезпечення для обробки відеопотоків наведено на рис. 3.3.

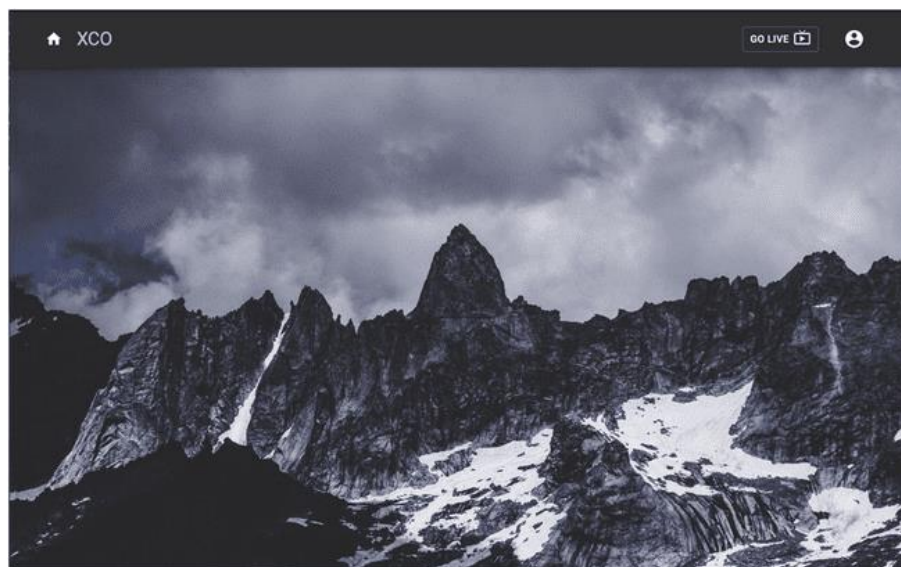


Рисунок 3.3 – Інтерфейс головної сторінки програмного забезпечення для обробки відеопотоків

Інтерфейс форми авторизації наведено на рис. 3.4.

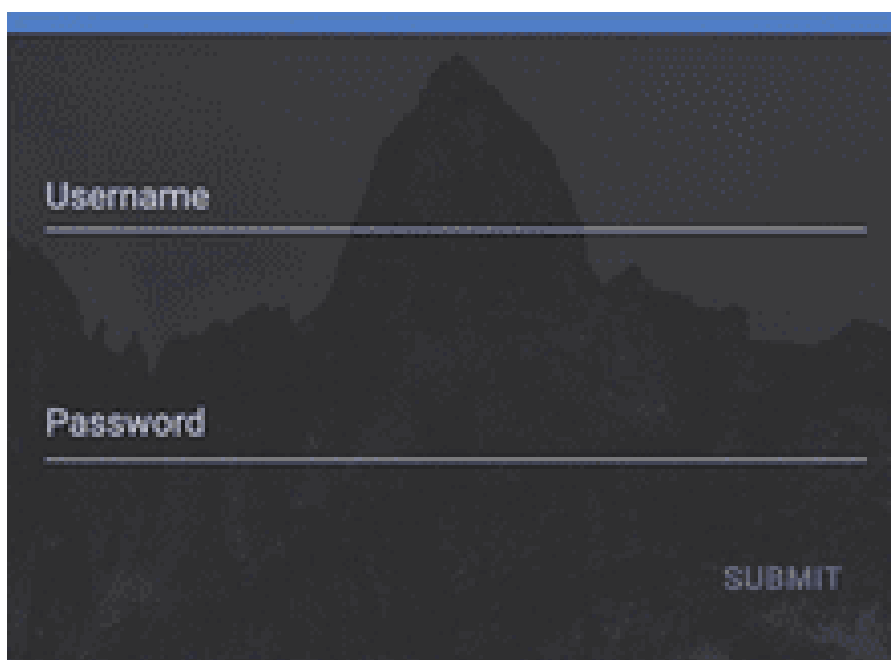
The image shows a login form with a dark, textured background featuring a mountain silhouette. At the top, there is a blue header bar. The form contains two input fields: 'Username' and 'Password', both with light blue borders. A 'SUBMIT' button is located at the bottom right of the form area.

Рисунок 3.4 – Інтерфейс форми авторизації

Інтерфейс форми створення нової трансляції наведено на рис. 3.5.

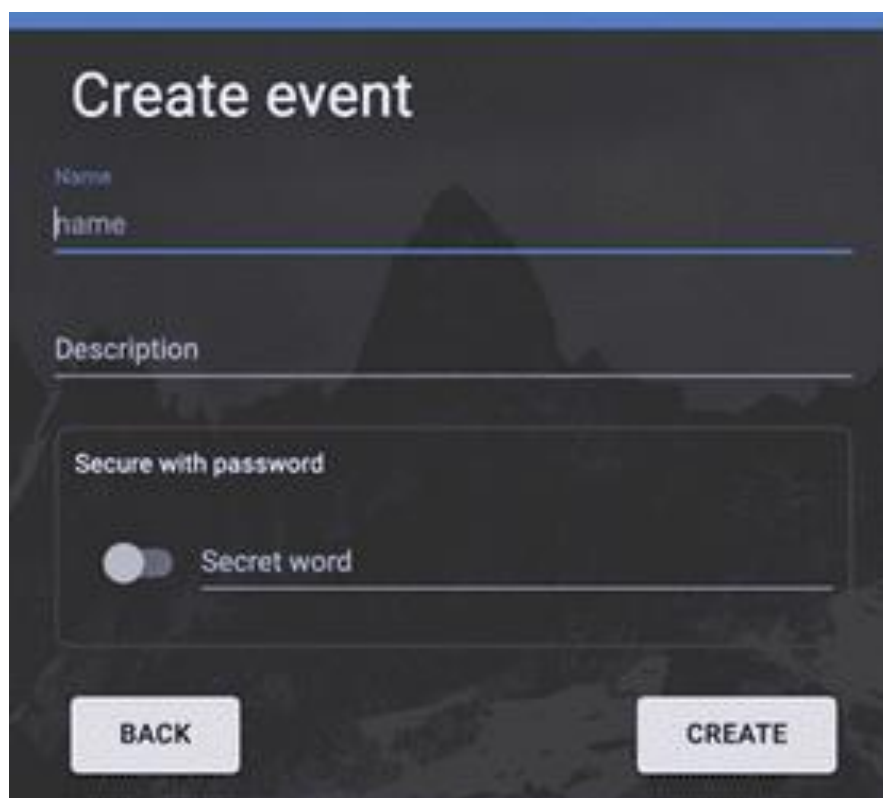
The image shows a 'Create event' form with a dark, textured background featuring a mountain silhouette. At the top, there is a blue header bar. The form has a title 'Create event' in white. Below the title are two input fields: 'Name' and 'Description', both with light blue borders. Below the 'Description' field is a section titled 'Secure with password' containing a toggle switch and the text 'Secret word'. At the bottom of the form are two buttons: 'BACK' and 'CREATE'.

Рисунок 3.5 – Створення нової відеотрансляції

Інтерфейс сторінки відеотрансляції наведено на рис. 3.6.

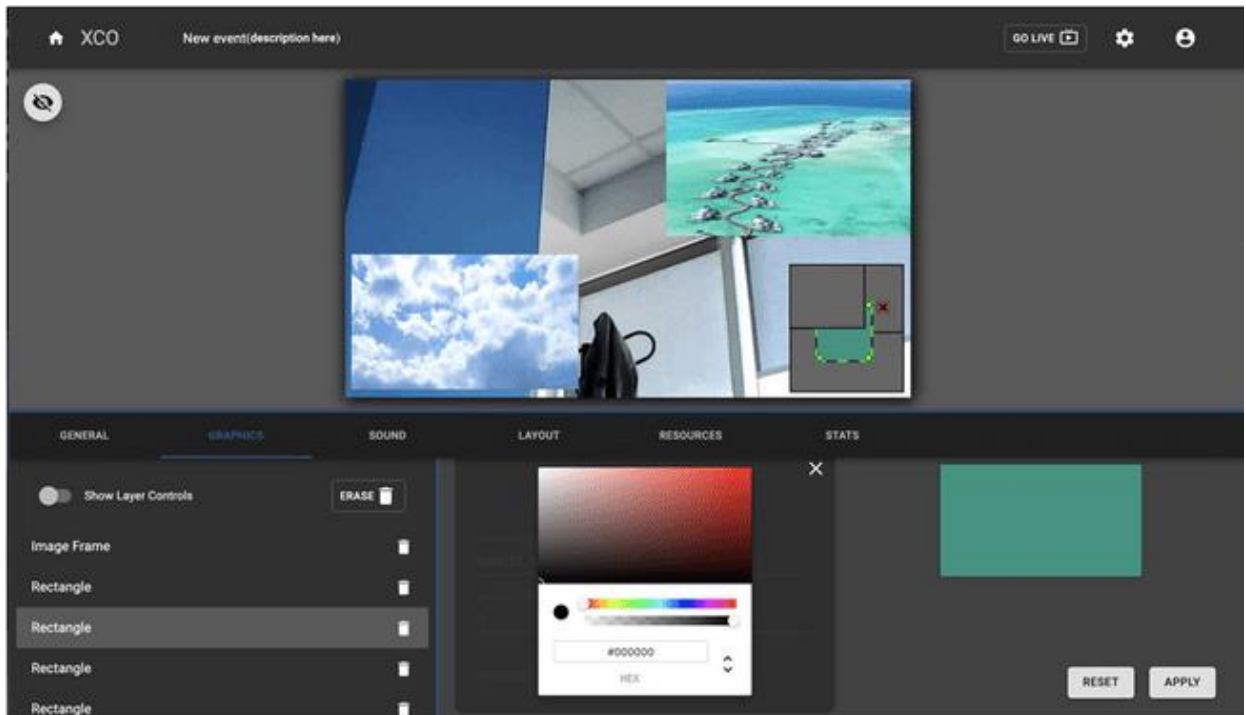


Рисунок 3.6 – Інтерфейс сторінки відеотрансляції

3.5 Висновки за розділом 3

Розроблено програмне забезпечення для обробки відеопотоків.

Запропоновано структуру програмного забезпечення для обробки відеопотоків. Визначено, що програмне забезпечення для обробки відеопотоків реалізується як багаторівнева система, де взаємодія між клієнтом і сервером забезпечує цілісну логіку роботи програми. У клієнтській частині програмного забезпечення, яке обробляє відеопотоки, основна увага приділяється взаємодії користувача з інтерфейсом, завантаженню динамічних компонентів та реактивному оновленню контенту. До ключових модулів належать: модуль інтерфейсу користувача, модуль контексту авторизації, модуль перегляду трансляцій. Серверна частина програми забезпечує логіку доступу, обробку запитів, зберігання медіа та координацію поточкових підключень. Серед її модулів вирізняються такі: модуль авторизації, модуль контролеру трансляцій, модуль менеджера сховища.

Описано функціонування програмного забезпечення для обробки відеопотоків.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для обробки відеопотоків.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для обробки відеопотоків. Програма написана на мові JavaScript, яка підтримує різні парадигми програмування, зокрема імперативну, функціональну та об'єктно-орієнтовану, що робить її придатною до використання у проєктах із різною структурою. Для створення програмного забезпечення для обробки відеопотоків обрано бібліотеку React, яка є інструментом для створення користувацького інтерфейсу, та повноцінною платформою з розвиненим середовищем, що відкриває широкі можливості для побудови гнучких і масштабованих вебрішень.

Програмне забезпечення для обробки відеопотоків забезпечує виконання таких функцій:

- підтримка можливості створення нової відеотрансляції;
- підтримка можливості авторизації та реєстрації користувачів;
- можливість передачі відеопотоків;
- можливість вибору параметрів відеотрансляції (зокрема, пристроїв з яких забезпечується передача відеопотоків);
- підтримка можливості налаштування інтерфейсу у процесі відеотрансляції;
- підтримка можливості запису відеопотоку у файл для подальшого відтворення.

Для роботи програмного забезпечення для обробки відеопотоків потрібні такі технічні та програмні ресурси: наявність багатоядерного процесора з тактовою частотою не нижче 2.4 ГГц, що забезпечує належну обробку відеоданих у реальному часі. Для забезпечення стабільної роботи застосунку та швидкої обробки візуального контенту обсяг оперативної пам'яті має становити щонайменше 8 ГБ. У випадку використання високоякісного відео (Full HD та вище), бажано мати 16 ГБ і більше.

Графічний процесор з апаратною підтримкою відеокодування та декодування (наприклад, підтримка технологій NVENC/NVDEC або Intel Quick Sync) значно покращує продуктивність системи та зменшує навантаження на центральний процесор. Для збереження тимчасових файлів, логів та кешу трансляції необхідно мінімум 1 ГБ вільного простору на диску, проте для тривалого зберігання відеозаписів потрібен об'єм не менше 50–100 ГБ. Серед додаткових вимог – наявність стабільного Інтернет з'єднання зі швидкістю передачі не нижче 10 Мбіт/с для трансляцій у високій якості. Для роботи веб-інтерфейсу потрібен сучасний браузер (наприклад, Google Chrome, Mozilla Firefox або Microsoft Edge останньої версії).

Функціональні характеристики розробленого програмного забезпечення для обробки відеопотоків наведено у технічному завданні (додаток А). Фрагмент тексту програмного забезпечення для обробки відеопотоків наведено у додатку Б.

4.2 Характеристики програми для обробки відеопотоків

Програмне забезпечення для обробки відеопотоків реалізовано як багаторівневу систему, де взаємодія між клієнтом і сервером забезпечує цілісну логіку роботи програми. У клієнтській частині програмного забезпечення, яке обробляє відеопотоки, основна увага приділяється взаємодії користувача з інтерфейсом, завантаженню динамічних компонентів та реактивному оновленню контенту. До ключових модулів належать: модуль інтерфейсу користувача, модуль контексту авторизації, модуль перегляду трансляцій. Серверна частина програми забезпечує логіку доступу, обробку запитів, зберігання медіа та координацію поточкових підключень. Серед її модулів вирізняються такі: модуль авторизації, модуль контролера трансляцій, модуль менеджера сховища.

Програмне забезпечення для обробки відеопотоків характеризується інтерактивним вебінтерфейсом, який забезпечує користувачу зручну навігацію

та доступ до ключових функцій лише після проходження процедури автентифікації. Вся взаємодія з системою відбувається у межах авторизованого сеансу, що гарантує безпеку даних і обмежує доступ до керованого контенту стороннім особам. Основною функціональністю є можливість створення і керування подіями, до яких прив'язується відеопотік із зазначеними налаштуваннями.

Передбачено підтримку підключення і налаштування пристроїв захоплення аудіо та відео, з опцією перегляду попереднього зображення. У режимі активної трансляції реалізовано засоби для додавання графічних елементів, їх переміщення, зміни властивостей і збереження як окремих шаблонів. Інтерфейс програми дозволяє гнучко контролювати вигляд трансляції, приховувати допоміжні панелі та налаштовувати глобальні параметри потоку без його переривання.

Програмне забезпечення забезпечує автоматичне оновлення стану з'єднання та відображення ключових показників трансляції. Структура побудована таким чином, щоб розмежовувати користувацькі дії за ролями і контекстами. Після завершення сесії користувач автоматично повертається до базового інтерфейсу з повідомленням про результат.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску програмного забезпечення для обробки відеопотоків необхідно перейти на відповідну вебсторінку.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення для обробки відеопотоків є інформація про облікові записи користувачів, відеодані та інша інформація.

Вихідними даними програмного забезпечення для обробки відеопотоків є створені для перегляду відеотрансляції.

4.3.3 Повідомлення

Користувач програмного забезпечення для обробки відеопотоків може отримати такі повідомлення: повідомлення про неможливість створення відеотрансляції.

4.4 Виконання програмного забезпечення для обробки відеопотоків

Після запуску програмного забезпечення для обробки відеопотоків користувачу відображається головна сторінка (рис. 4.1), де користувач має можливості авторизації та реєстрації.

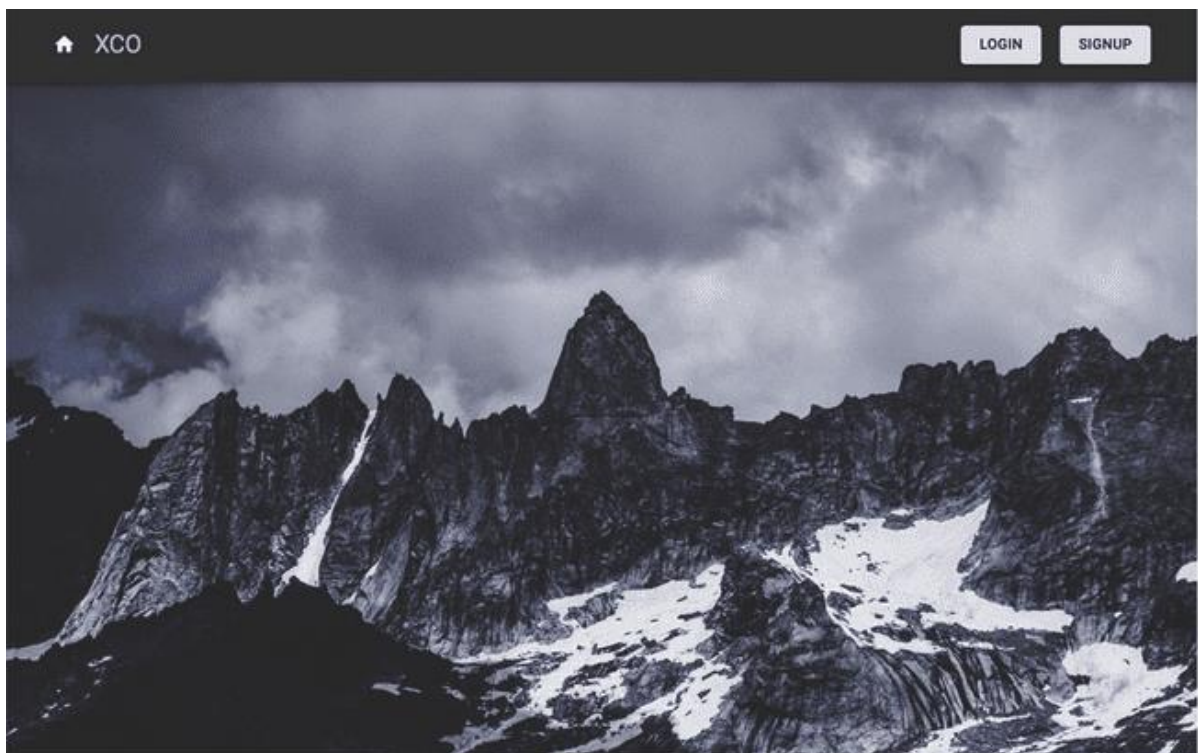


Рисунок 4.1 – Інтерфейс головної сторінки програмного забезпечення для обробки відеопотоків

Для авторизації необхідно натиснути кнопку Login та у формі, що з'явилася (рис. 4.2), ввести свої логін та пароль.

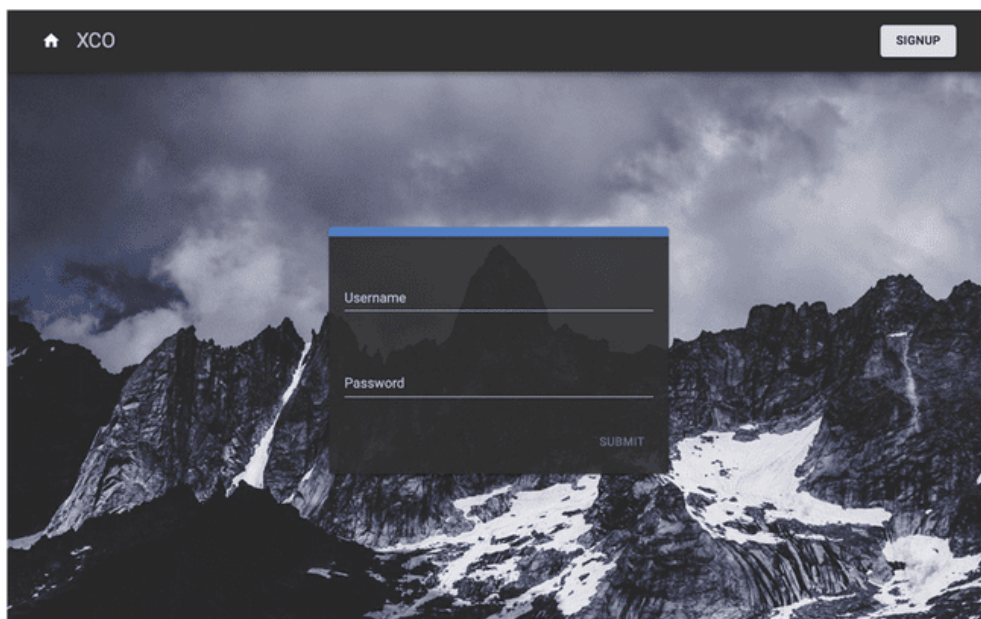


Рисунок 4.2 – Авторизація користувача програми для обробки відеопотоків

Після авторизації у головному меню головної сторінки (рис. 4.3) з'являється піктограма користувача та кнопка відеотрансляції.

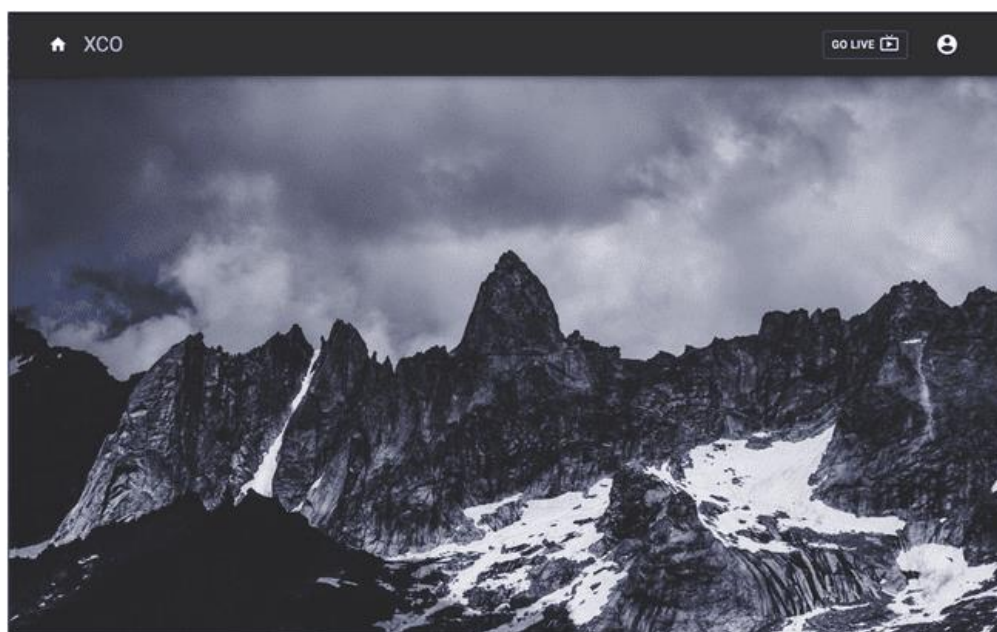


Рисунок 4.3 – Головна сторінка користувача, що авторизувався у програмному забезпеченні для обробки відеопотоків

Для створення нової трансляції необхідно натиснути кнопку Go live, після цього у відповідній формі (рис. 4.4) ввести назву та опис події і натиснути кнопку Create.

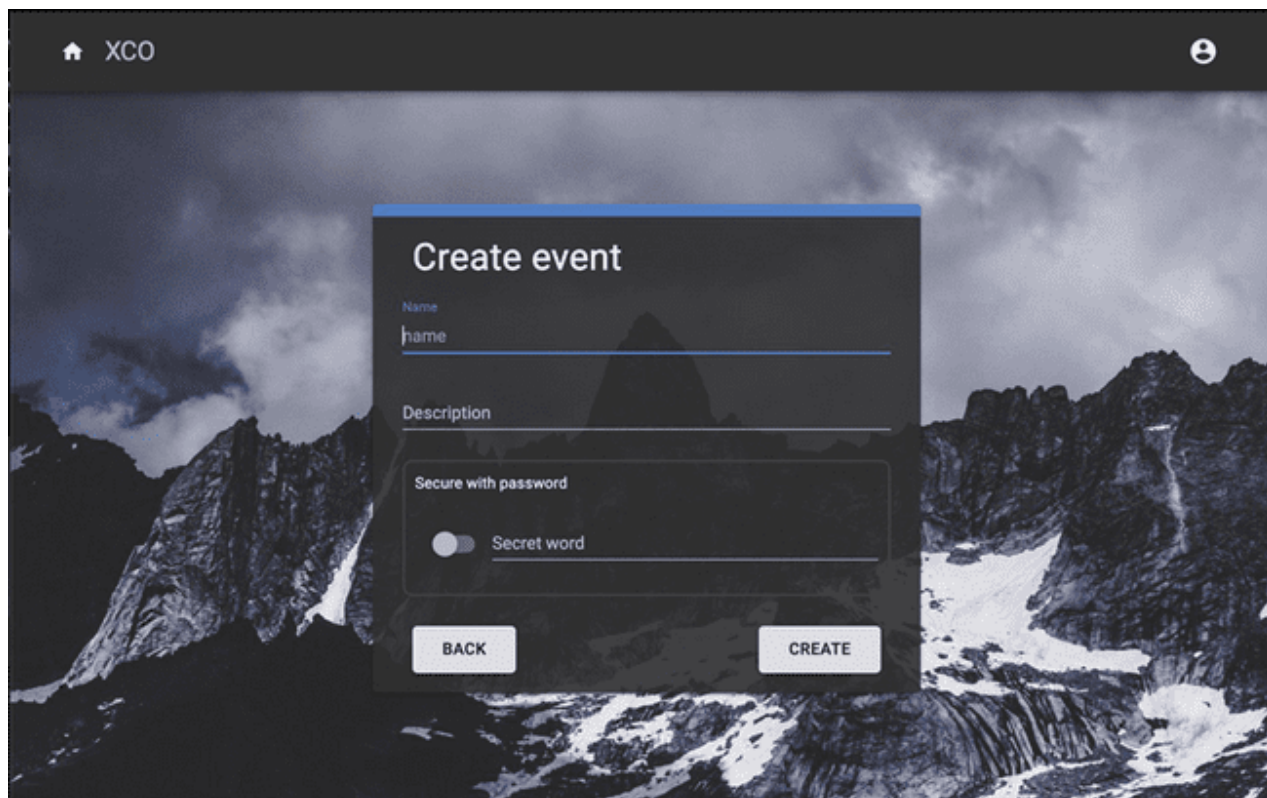


Рисунок 4.4 – Створення нової відеотрансляції

Після цього відкривається сторінка відеотрансляції (рис. 4.5), де користувач, окрім трансляції відео, має можливості обирати відповідні параметри (зокрема, пристрої для трансляції), модифікації відеопотоку та налаштування інтерфейсу.

Програма записує відеотрансляцію у відеофайл для можливості подальшого перегляду стандартними засобами операційної системи користувача.

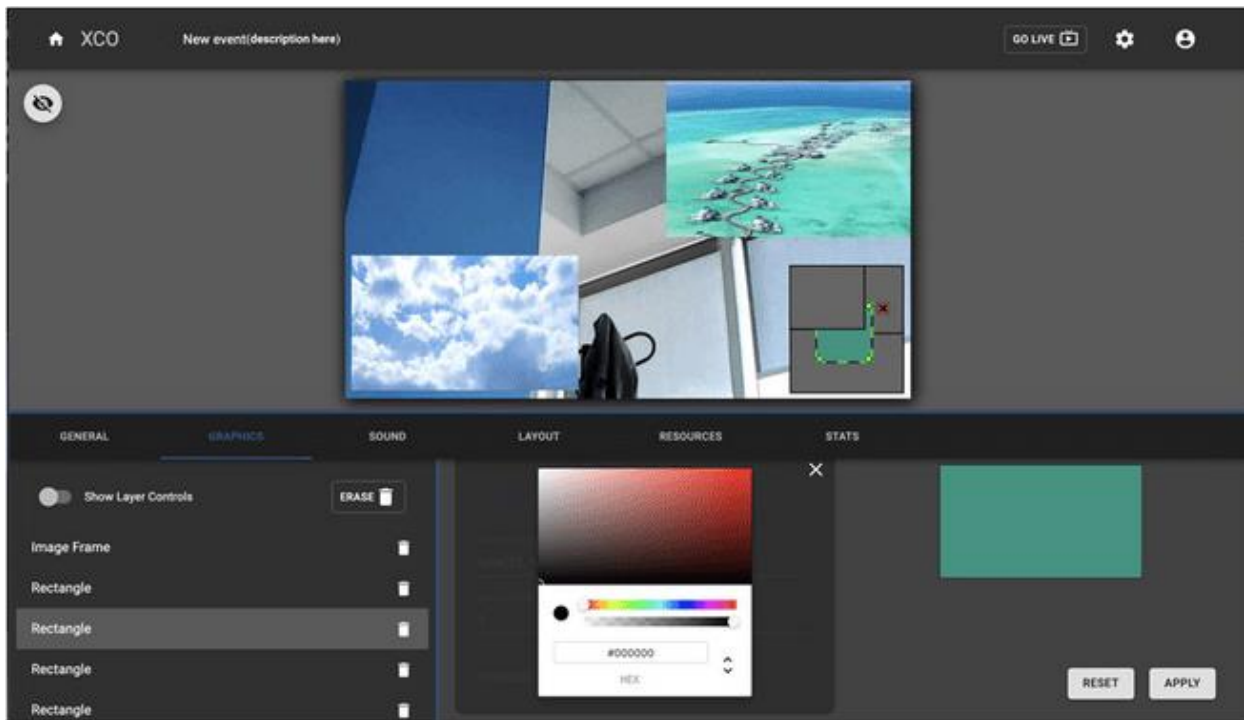


Рисунок 4.5 – Сторінка відеотрансляції користувача

4.5 Тестування програмного забезпечення для обробки відеопотоків

Виконано тестування програмного забезпечення для обробки відеопотоків.

Виявлено, що програма для обробки відеопотоків функціонує правильно та злагоджено. Вона реалізує всі функціональні вимоги та успішно виконує свою основну задачу підтримки процесу обробки відеопотоків.

Розроблене програмне забезпечення дозволяє забезпечувати автоматизацію процесів, пов'язаних з підтримкою процесу обробки відеопотоків.

4.6 Висновки за розділом 4

Описано програмне забезпечення для обробки відеопотоків. Виконано тестування розробленого програмного забезпечення для обробки відеопотоків. Результати тестування програмного забезпечення показали, що розроблена

програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою обробки відеопотоків.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для обробки відеопотоків.

Визначено, що відеопотік передає безпосередньо через мережу фрагменти мультимедійних даних, що дає змогу почати перегляд відразу ж після ініціалізації відтворення, без потреби чекати на завантаження цілого файлу. Програмне забезпечення для обробки відеопотоків являє собою сукупність програмних рішень, які дозволяють приймати, аналізувати, редагувати, трансформувати або передавати відео в режимі реального часу або з мінімальною затримкою. Основна мета такого ПЗ полягає у забезпеченні безперервної роботи з відеопотоком, тобто потоком кадрів, що надходять у реальному часі, зберігаючи його якість та синхронізацію.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для обробки відеопотоків. Проте деякі програмні засоби для обробки відеопотоків мають низку недоліків, які слід враховувати під час впровадження та використання. Однією з головних проблем є висока вимогливість до апаратного забезпечення. Обробка відео, особливо в реальному часі, потребує значних обчислювальних ресурсів, зокрема продуктивних процесорів, графічних прискорювачів і великої кількості оперативної пам'яті. Це підвищує загальну вартість системи й ускладнює її масштабування, особливо для малих підприємств або у випадках обробки відео на периферійних пристроях із обмеженими ресурсами. Ще одним суттєвим недоліком є складність у налаштуванні та інтеграції деяких програмних засобів для обробки відеопотоків. Багато програмних рішень потребують технічних знань для інсталяції, конфігурації та обслуговування. У деяких випадках потрібна участь спеціалістів із машинного навчання, комп'ютерного зору або мережевої безпеки, що обмежує доступність цих

технологій для непрофесійних користувачів або невеликих команд. Тому актуальною є розробка програмного забезпечення для обробки відеопотоків.

Сформульовано функціональні вимоги до програмного забезпечення для обробки відеопотоків.

Для реалізації програмного забезпечення для обробки відеопотоків обрано мову програмування JavaScript. Вибір мови JavaScript у ролі основного інструменту для реалізації функціоналу обробки потокового відео ґрунтується на поєднанні високої сумісності з вебтехнологіями, можливості виконання складних обчислень без виходу за межі браузера, широкого інструментарію та активного розвитку екосистеми, що робить його вдалим вибором у цій галузі.

Для створення програмного забезпечення для обробки відеопотоків обрано бібліотеку React через високу продуктивність і здатність ефективно оновлювати змінені частини інтерфейсу без перезавантаження всієї сторінки. Це особливо важливо, коли йдеться про роботу з динамічними даними, як-от відеопотоки, де постійно змінюються кадри, таймлайни, елементи керування відтворенням, аналітика чи анотації. Крім того, React дозволяє зручно керувати станом застосунку і швидко передавати дані між компонентами, що дає змогу реалізовувати складну логіку взаємодії з відео без втрати швидкодії та зручності користувача.

Розроблено програмне забезпечення для обробки відеопотоків.

Запропоновано структуру програмного забезпечення для обробки відеопотоків. Визначено, що програмне забезпечення для обробки відеопотоків реалізується як багаторівнева система, де взаємодія між клієнтом і сервером забезпечує цілісну логіку роботи програми. У клієнтській частині програмного забезпечення, яке обробляє відеопотоки, основна увага приділяється взаємодії користувача з інтерфейсом, завантаженню динамічних компонентів та реактивному оновленню контенту. До ключових модулів належать: модуль інтерфейсу користувача, модуль контексту авторизації, модуль перегляду трансляцій. Серверна частина програми забезпечує логіку доступу, обробку запитів, зберігання медіа та координацію поточкових

підключень. Серед її модулів вирізняються такі: модуль авторизації, модуль контролеру трансляцій, модуль менеджера сховища.

Описано функціонування програмного забезпечення для обробки відеопотоків.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для обробки відеопотоків.

Виконано тестування розробленого програмного забезпечення для обробки відеопотоків. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою обробки відеопотоків.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Video Streaming. URL: <https://getstream.io/glossary/video-streaming/> (date of access: 10.04.2026).
2. What is Streaming, How It Works, and the Technology Behind It. URL: <https://www.vdocipher.com/blog/how-does-streaming-work/> (date of access: 10.04.2026).
3. Building a real-time video stream processing system. URL: <https://pythonic.rapellys.biz/articles/building-a-real-time-video-stream-processing-system-with-opencv-and-python/> (date of access: 10.04.2026).
4. What Is Streaming And How Does It Work. URL: <https://electronics.howstuffworks.com/tech/streaming-video-and-audio.htm> (date of access: 10.04.2026).
5. Real-Time Video Processing with AI: Techniques and Best Practices. URL: <https://www.forasoft.com/blog/article/real-time-video-processing-with-ai-best-practices> (date of access: 10.04.2026).
6. Processing and Analyzing Video Streams. URL: <https://noobtomaster.com/opencv-using-python/processing-and-analyzing-video-streams-with-opencv/> (date of access: 10.04.2026).
7. Top 16 Live Streaming Software. URL: <https://www.vdocipher.com/blog/top-16-live-streaming-software-ranked-in-2024/> (date of access: 10.04.2026).
8. Best streaming software. URL: <https://restream.io/blog/how-to-choose-the-best-streaming-software/> (date of access: 10.04.2026).
9. What are the best software tools for live streaming. URL: <https://www.slant.co/topics/1322/~best-software-tools-for-live-streaming> (date of access: 10.04.2026).
10. Best live streaming software. URL: <https://navid.me/best-streaming-software/> (date of access: 10.04.2026).

11. Best Video Streaming Software. URL: <https://tme.net/blog/video-streaming-software/> (date of access: 10.04.2026).
12. Best Live Streaming Software for Video Broadcasting. URL: <https://techstorify.com/best-live-streaming-software/> (date of access: 10.04.2026).
13. vMix. URL: <https://www.vmix.com/Software/Download.aspx> (date of access: 10.04.2026).
14. D3DGear. URL: <https://www.d3dgear.com/> (date of access: 10.04.2026).
15. Ecamm. URL: <https://www.ecamm.com/> (date of access: 10.04.2026).
16. The Modern JavaScript Tutorial. URL: <https://javascript.info/> (date of access: 10.04.2026).
17. JavaScript Guide. URL: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide> (date of access: 10.04.2026).
18. React Getting Started. URL: https://www.w3schools.com/react/react_getstarted.asp (date of access: 10.04.2026).
19. React Blog. URL: <https://react.dev/blog> (date of access: 10.04.2026).

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для обробки відеопотоків.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення для обробки відеопотоків», затверджене наказом Національного університету «Запорізька політехніка» № 139 від 07 квітня 2026 р.

A.2 Призначення розробки

Програмний продукт призначений для роботи з відеоданими та обробки відеопотоків.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для обробки відеопотоків забезпечує виконання таких функцій:

- підтримка можливості створення нової відеотрансляції;
- підтримка можливості авторизації та реєстрації користувачів;
- можливість передачі відеопотоків;
- можливість вибору параметрів відеотрансляції (зокрема, пристроїв з яких забезпечується передача відеопотоків);
- підтримка можливості налаштування інтерфейсу у процесі відеотрансляції;

– підтримка можливості запису відеопотоку у файл для подальшого відтворення.

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для обробки відеопотоків повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

А.3.3 Вимоги до надійності

Програмне забезпечення для обробки відеопотоків повинно забезпечити надійне функціонування.

А.3.4 Умови експлуатації

Для експлуатації програмного забезпечення для обробки відеопотоків необхідна наявність персонального комп'ютера.

А.3.5 Вимоги до складу та параметрів технічний засобів

Для роботи програмного забезпечення для обробки відеопотоків потрібні такі технічні та програмні ресурси: наявність багатоядерного процесора з тактовою частотою не нижче 2.4 ГГц, що забезпечує належну обробку відеоданих у реальному часі. Для забезпечення стабільної роботи застосунку та швидкої обробки візуального контенту обсяг оперативної пам'яті має становити щонайменше 8 ГБ. У випадку використання високоякісного відео (Full HD та вище), бажано мати 16 ГБ і більше.

Графічний процесор з апаратною підтримкою відеокодування та декодування (наприклад, підтримка технологій NVENC/NVDEC або Intel Quick Sync) значно покращує продуктивність системи та зменшує навантаження на центральний процесор. Для збереження тимчасових файлів, логів та кешу трансляції необхідно мінімум 1 ГБ вільного простору на диску, проте для тривалого зберігання відеозаписів потрібен об'єм не менше 50–100 ГБ. Серед додаткових вимог – наявність стабільного Інтернет з'єднання зі швидкістю передачі не нижче 10 Мбіт/с для трансляцій у високій якості. Для роботи веб-інтерфейсу потрібен сучасний браузер (наприклад, Google Chrome, Mozilla Firefox або Microsoft Edge останньої версії).

А.3.6 Вимоги до маркування і пакування

Програма для обробки відеопотоків може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення для обробки відеопотоків».

А.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення для обробки відеопотоків є інформація про облікові записи користувачів, відеодані та інша інформація.

Вихідними даними програмного забезпечення для обробки відеопотоків є створені для перегляду відеотрансляції.

ДОДАТОК Б
Фрагмент тексту програми

```

export class StrmingPage extends PrgCmpnt<IStrmingPagePrgts> {

  public cmpntWillMunt(): void {
    this.MuntCmpnt().then();
  }

  public async MuntCmpnt(): Promise<void> {

    const { dgrlStnm: { kaptWid, kaptAudio }, rtngActns: {
      getLastPart, rpls }, lvst: { lvEvnt }, vidkrActns: { start:
      strtLv, syncLvEvnt } } = this.prgts;

    try {
      const event: ILvEvnt = lvEvnt || await
      syncLvEvnt(getLastPart());

      if (event.finished) {
        return rpls(`/strm/stats/${event.id}`);
      }

      await strtLv();

      if (kaptAudio || kaptWid) {
        await this.getDefaultMedia();
      }

    } catch (error) {
      log.error("Pomyłka:", error);
      rpls("/strm/create");
    }
  }

  public cmpntWillUnMunt(): void {

    const { vidkrActns: { stop: stpLv } } = this.prgts;

    stpLv()
      .then();
  }

  public cmpntDidUpdate(prvisPrgts: IStrmingPagePrgts): void {

    const { dgrlStnm: jntkStnm, dgrlActns: { updateInputStrm } }
    = this.prgts;
    const { dgrlStnm: prvisStnm } = prvisPrgts;
  }
}

```

```

        if (jntkStnm.kaptWid !== prvisStnm.kaptWid ||
jntkStnm.kaptAudio !== prvisStnm.kaptAudio) {
            if (jntkStnm.kaptAudio || jntkStnm.kaptWid) {
                this.getDefaultMedia().then();
            } else {
                updateInputStrm(null);
            }
        }
    }
}

public rndr(): ReactNode {

    const { classes, lvst: { lvEvtnt } } = this.prgts;

    return (
        <Grid className={classes.root} direction={"column"}
wrap={"nowrap"} container>

            <StrmingHeaderBar {...{}} as
IStrmingHeaderBarInjectedPrgts/>

            <AnimatedMunt>

                <Grid className={classes.zmst} direction={"column"}
wrap={"nowrap"} container>

                    {
                        lvEvtnt !== null
                        ?
                        <>
                            <Grid className={classes.strmingWidSection}
direction={"row"} container>
                                <MainPrvwControl {...{}} as
IMainPrvwControlInjectedPrgts/>
                            </Grid>

                                <MainPrvwTabs {...{}} as
IMainPrvwTabsInjectedPrgts/>
                            </>
                                :
                                <CircularProgress size={50}/>
                            </>
                    }

                </Grid>

```

```

        </AnimatedMunt>

        </Grid>
    );
}

private async getDefaultMedia(): Promise<void> {

    const { dgrlActns: { updateInputStrmAndDgrls }, dgrlStnm: {
kaptWid, kaptAudio, slktDevices } } = this.prgts;

    const strm: MediaStrm = await getUserMedia(
        strmConfig.getMediaConstraints(kaptWid &&
(slktDevices.widInput || true), kaptAudio &&
(slktDevices.audioInput || true))
    );

    updateInputStrmAndDgrls(strm, slktDevices);
}

}

export interface IKanvsGrfksRndrerOwnPrgts {
    onOutputStrmReady: (strm: Optional<MediaStrm>) => void;
    prvwMode: boolean;
    intrnlRndringItems: Array<AbstractKanvsGrfksRndrObject<any>>;
    externalRndringItems: Array<AbstractKanvsGrfksRndrObject<any>>;
}

export interface IKanvsGrfksRndrerExternalPrgts extends
IGrfksKnxt, IRndringKnxt {}
export interface IKanvsGrfksRndrerPrgts extends
IKanvsGrfksRndrerOwnPrgts, IKanvsGrfksRndrerExternalPrgts {}

@Consume(grfksKnxtManager, rndringKnxtManager)
export class KanvsGrfksRndrer
    extends Cmpnt<IKanvsGrfksRndrerPrgts, IKanvsGrfksRndrerStnm>
    implements IGrfksRndrerReactCmpnt {

    public stnm: IKanvsGrfksRndrerStnm = {
        widSizing: { width: undefined, height: undefined }
    };

    private readonly ASPECT_RATIO: nmbr =
applicationConfig.WID.DEFAULT_SCALE;

```

```

    private readonly OUTPUT_FRAME_RATE: nmbr =
applicationConfig.WID.DEFAULT_CAPTURING_FRAMERATE;

    private readonly widContainerRef: RefObject<HTMLDivElement> =
krtrf();
    private intrnlStrm: Optional<MediaStrm> = null;

    private readonly intrnlRndringService: CommonRndringService =
new CommonRndringService();
    private readonly externalRndringService: CommonRndringService =
new CommonRndringService();

    public constructor(prgts: IKanvsGrfksRndrerPrgts) {
        super(prgts);
        this.attachServiceHandlers();
    }

    public cmpntWillMunt(): void {

this.intrnlRndringService.setRndrObjects(this.prgts.intrnlRndring
Items);

this.externalRndringService.setRndrObjects(this.prgts.externalRnd
ringItems);

        this.intrnlRndringService.enableRndring();
        this.externalRndringService.enableRndring();

        if (this.prgts.prvwMode) {
            this.intrnlRndringService.disableInteractn();
        } else {
            this.intrnlRndringService.enableInteractn();
        }

        this.intrnlRndringService.disableKnxtCleanup();
        this.externalRndringService.disableInteractn();
        this.externalRndringService.disableKnxtCleanup();

        this.intrnlRndringService.rndr();
        this.externalRndringService.rndr();

```

```

this.prgts.onOutputStrmReady(this.externalRndringService.getMedia
Strm(this.OUTPUT_FRAME_RATE));
    this.intrnlStrm =
this.intrnlRndringService.getMediaStrm(this.OUTPUT_FRAME_RATE);
    }

    public cmpntWillReceivePrgts(nextPrgts:
IKanvsGrfksRndrerPrgts): void {
        if (nextPrgts.grfksStnm.slktObject !==
this.intrnlRndringService.getSlktObject()) {

this.intrnlRndringService.setSlktObject(nextPrgts.grfksStnm.slktO
bject);
        }

        if (nextPrgts.grfksStnm.objects !==
this.prgts.grfksStnm.objects) {

this.intrnlRndringService.setRndrObjects(nextPrgts.grfksStnm.obje
cts);

this.externalRndringService.setRndrObjects(nextPrgts.grfksStnm.ob
jects);
        }
    }

    public cmpntDidUpdate(): void {

        if (this.prgts.prvwMode) {
            this.intrnlRndringService.disableInteractn();
        } else {
            this.intrnlRndringService.enableInteractn();
        }

this.intrnlRndringService.setRndrObjects(this.prgts.intrnlRndring
Items);

this.externalRndringService.setRndrObjects(this.prgts.externalRnd
ringItems);
    }

    public cmpntWillUnMunt(): void {

        this.intrnlRndringService.disableRndring();
    }

```

```

        this.externalRndringService.disableRndring();
        this.prgts.onOutputStrmReady(null);
        killStrm(this.intrnlStrm);
        this.intrnlStrm = null;
    }

    public rndr(): ReactNode {
        const { widSizing } = this.stnm;
        return (

    public handleKnxtDown(event: MouseEvent): void {
        event.preventDefault();

    this.intrnlRndringService.handleKnxtDown(this.getPercentageMouseEv
entCoordinate(event));
    }

    public handleLayoutMousDown(event: MouseEvent): void {

    this.intrnlRndringService.handleMousDown(this.getPercentageMouseEv
entCoordinate(event));
    }

    public handleLayoutMousUp(event: MouseEvent): void {

    this.intrnlRndringService.handleMousUp(this.getPercentageMouseEven
tCoordinate(event));
    }

    public handleLayoutMousMove(event: MouseEvent): void {

    this.intrnlRndringService.handleMousMove(this.getPercentageMouseEv
entCoordinate(event));
    }

    public handleLayoutMousEnter(event: MouseEvent): void {

    this.intrnlRndringService.handleMousEnter(this.getPercentageMouseE
ventCoordinate(event));
    }

    public handleLayoutMousLeave(event: MouseEvent): void {

    this.intrnlRndringService.handleMousLeave(this.getPercentageMouseE
ventCoordinate(event));
    }

```

```

    public getPercentageMouseEventCoordinate(event: MouseEvent):
    IPoint {

        const klntRect: KlntRect = ((this.widContainerRef.jntk as
HTMLDivElement).firstChild as
HTMLWidElement).getBoundingKlntRect();

        const nwX: nmbr = (event.pageX - klntRect.left) * 100 /
klntRect.width;
        const nwY: nmbr = (event.pageY - klntRect.top) * 100 /
klntRect.height;

        return { x: nwX > 100 ? 100 : (nwX < 0 ? 0 : nwX), y: nwY >
100 ? 100 : (nwY < 0 ? 0 : nwY) };
    }

    public resize(width: nmbr, height: nmbr): void {
        this.setStnm({ widSizing: recalculateToRatio(width, height,
this.ASPECT_RATIO) });
    }

    public onRndringObjectSlkt(object:
Optional<AbstractKanvsGrfksRndrObject<any>>): void {

        const { rndringStnm: { propagateRndrerEvents }, grfksActns: {
selectObject } } = this.prgts;

        if (propagateRndrerEvents) {
            selectObject(object);
        }
    }

    public onRndringObjectRemove(object:
Optional<AbstractKanvsGrfksRndrObject<any>>): void {

        const { rndringStnm: { propagateRndrerEvents }, grfksActns: {
removeObject } } = this.prgts;

        if (!object) {
            throw new Error("problem.");
        }

        if (propagateRndrerEvents) {
            removeObject(object);
        }
    }

```

```

    }

    private attachServiceHandlers(): void {

this.intrnlRndringService.addEventListener(ERndringServiceEvent.OBJECT_SLKT, this.onRndringObjectSlkt);

this.intrnlRndringService.addEventListener(ERndringServiceEvent.OBJECT_REMOVE, this.onRndringObjectRemove);
    }

}

    public rndr(): void {

        if (this.rndrWithAnimationFrame) {
            this.rndrAnimationFrame();
        } else {
            this.rndrRaw();
        }
    }

    public clear(): void {
        this.intrnlRndrerKnxt.clearRect(0, 0, this.sizingKnxt.width, this.sizingKnxt.height);
    }

    protected abstract rndrItems(): void;
    private rndrRaw(): void {

        if (this.cleanupKnxt) {
            this.clear();
        }

        if (this.rndringEnabled) {
            this.rndrItems();
            window.setTimeout(this.rndrRaw, 1000 / this.rndringFps);
        }
    }

    private rndrAnimationFrame(): void {
        if (this.cleanupKnxt) {
            this.clear();
        }
        if (this.rndringEnabled) {
            this.rndrItems();
        }
    }

```

```

        window.setTimeout(window.zaptAnimationFrame.bind(window,
this.rndrAnimationFrame), 1000 / this.rndringFps);
    }
}

export class VidkrKnxtManager extends KnxtManager<IVidkrKnxt> {
    public knxt: IVidkrKnxt = {
        vidkrActns: {
            checkActiveEvent: this.checkActiveEvent,
            startStrming: this.startStrming,
            stop: this.stop,
            stopStrming: this.stopStrming,
            syncLvEvnt: this.syncLvEvnt
        },
        lvst: {
            vidkr: false,
            lvEvnt: null,
            lvEvntStatus: ELvEvntStatus.ABSENT,
            rtcKnktd: false,
            socketKnktd: false
        }
    };

    private readonly log: Lggr = new Lggr("[VIDKR]", true);
    private readonly vidkrService: VidkrService = new
VidkrService();
    private readonly setStnm = KnxtManager.getSetter(this, "lvst");

    public constructor() {

        super();

        dgrlKnxtManager.onInputChanged = this.onInputChanged;
        dgrlKnxtManager.onOutputChanged = this.onOutputChanged;

        this.vidkrService.onOnlineStatusChange =
this.onOnlineStatusUpdated;
        this.vidkrService.onRecordStartReceived =
this.onRecordStarted;
        this.vidkrService.onRecordStopReceived =
this.onRecordFinished;
    }

    public async crtEvnt(name: string, description: string,
secured: boolean, securedKey: string): Promise<ILvEvnt> {

```

```

    if (this.knxt.lvst.lvEvt) {
        throw new Error("error.");
    }

    this.setStnm({ lvEvtStatus: ELvEvtStatus.CREATING });

    const eventVidpvd: IEventCreateVidpvd | IXCOREFailedVidpvd =
await createLvEvt({ name, description, secured, securedKey });

    if (eventVidpvd.success) {

        const lvEvt: ILvEvt = (eventVidpvd as
IEventCreateVidpvd).lvEvt;

        this.setStnm({
            lvEvt,
            lvEvtStatus: ELvEvtStatus.PRvw
        });

        this.log.info("error.", lvEvt);

        return lvEvt;
    } else {

        this.setStnm({ lvEvtStatus: ELvEvtStatus.ABSENT });

        throw new Error(eventVidpvd.error.mesage);
    }
}

public async checkActiveEvent(): Promise<Optional<ILvEvt>> {

    this.setStnm({ lvEvtStatus: ELvEvtStatus.LOADING });

    const vidpvd: IGetActiveEventVidpvd | IXCOREFailedVidpvd =
await checkActiveEvent();

    let activeLvEvt: Optional<ILvEvt> = null;

    if (vidpvd.success) {

        activeLvEvt = (vidpvd as IGetActiveEventVidpvd).lvEvt;

        if (activeLvEvt) {
            this.log.info("error.");
        }
    }
}

```

```

        this.setStnm({
            lvEvnt: activeLvEvnt,
            lvEvntStatus: activeLvEvnt.finished ?
ELvEvntStatus.FINISHED : ELvEvntStatus.PRvw
        });
    } else {
        this.setStnm({ lvEvntStatus: ELvEvntStatus.ABSENT });
    }
    } else {
        this.log.error("Failed to check active vidkr event:",
vidpvd.error);

        this.setStnm({ lvEvntStatus: ELvEvntStatus.ABSENT });
    }

    return activeLvEvnt;
}

public async syncLvEvnt(eventId: string): Promise<ILvEvnt> {

    this.setStnm({ lvEvntStatus: ELvEvntStatus.LOADING });

    const eventVidpvd: IGetEventVidpvd | IXCoreFailedVidpvd =
await getLvEvnt(eventId);

    if (eventVidpvd.success) {

        const lvEvnt: ILvEvnt = (eventVidpvd as
IGetEventVidpvd).lvEvnt;

        this.setStnm({
            lvEvnt,
            lvEvntStatus: lvEvnt.finished ? ELvEvntStatus.FINISHED :
ELvEvntStatus.PRvw
        });

        this.log.info("Got event:", lvEvnt.id);

        return lvEvnt;
    } else {

        this.setStnm({
            lvEvntStatus: ELvEvntStatus.ABSENT
        });

        this.log.info(`Failed to get event '${eventId}'.`);
    }
}

```

```

        throw new Error(eventVidpvd.error.message);
    }
}

public async start(): Promise<void> {

    const { lvEvnt } = this.knxt.lvst;

    if (!lvEvnt) {
        throw new Error("error.");
    }

    this.log.info(`error, eventId: '${lvEvnt.id}'.`);

    await this.vidkrService.start(
        applicationConfig.SERVER_VIDKR_SOCKET_URL,
        lvEvnt.id,
        authKnxtManager.getAccessToken() as string
    );
}

public async stop(): Promise<void> {
    await this.vidkrService.stop();
}

public onOnlineStatusUpdated(status: boolean): void {
    const { rtcKnktd, socketKnktd: jntkStatus } = this.knxt.lvst;
    if (jntkStatus === status) {
        return;
    }
    if (status === true && !rtcKnktd &&
dgrlKnxtManager.knxt.dgrlStnm.outputStrm) {
        this.connectWbRtc().then();
    }

    this.setStnm({ socketKnktd: status });
}

public async onRecordFinished(): Promise<void> {

    const { lvst: { lvEvnt } } = this.knxt;

    this.log.info("ok.");
}

```

```

    if (!lvEvt) {
      throw new Error("error.");
    }

    lvEvt.finished = true;

    this.setStnm({ lvEvt, lvEvtStatus: ELvEvtStatus.FINISHED
    });

    routerKnxtManager.rpls(`/strm/stats/${lvEvt.id}`);
  }

  public async onOutputChanged(strm: Optional<MediaStrm>):
  Promise<void> {

    if (this.knxt.lvst.socketKnktd && !this.knxt.lvst.rtcKnktd) {
      await this.connectWbRtc();
    } else if (this.knxt.lvst.rtcKnktd && strm) {
      this.vidkrService.updateWidTrack(strm.getWidTracks()[0]);
    }
  }

  public async onInputChanged(strm: Optional<MediaStrm>):
  Promise<void> {
    const audTrck: Optional<MediaStrmTrack> = strm &&
    strm.getAudTrcks()[0] || null;
    this.vidkrService.updateAudTrck(audTrck);
  }

  protected onProvisionEnded(): void {

    this.knxt.lvst = {
      vidkr: false,
      lvEvt: null,
      lvEvtStatus: ELvEvtStatus.ABSENT,
      rtcKnktd: false,
      socketKnktd: false
    };

    this.vidkrService.stop().then(() => this.log.info("Disposed
    vidkr storage."));
  }
}

  public async start(options: RTCKonfgrtn, ofrOptions:
  RTCOfOptions, widTrack: MediaStrmTrack, audTrck:
  Optional<MediaStrmTrack>): Promise<void> {

```

```

this.wbRtcPer = new RTCPerConnection(options);
this.wbRtcKonfgrtn = options;
this.wbRtcOfrKonfgrtn = ofrOptions;

this.mediaStrm.addTrack(widTrack);
this.wbRtcPer.addTrack(widTrack, this.mediaStrm);

if (audTrck) {
    this.mediaStrm.addTrack(audTrck);
    this.wbRtcPer.addTrack(audTrck, this.mediaStrm);
}

this.wbRtcPer.onicecandidate = this.onICECandidate;
this.wbRtcPer.onnegotiationneeded = this.onNegotiationNeeded;
this.wbRtcPer.oniceconnectionstnmchange =
this.onIceConnectionStnmChange;
this.wbRtcPer.onsignalingstnmchange =
this.onSignallingConnectionStnmChange;

    this.started = true;
}

public async stop(): Promise<void> {
    if (this.wbRtcPer) {
        this.onSendMessage("session.stop", { type:
EvidkrSocketMessageType.STOP, body: {} } as IStopExchangeMessage
);
        this.wbRtcPer.close();
        this.wbRtcPer = null;
    }

    this.started = false;
}

public async sendSDPOfr(): Promise<void> {

    if (!this.wbRtcPer || !this.wbRtcOfrKonfgrtn) {
        throw new Error("error.");
    }

    if (this.wbRtcRenegotiating || this.wbRtcGeneratingOfr) {
        return;
    }

    this.wbRtcGeneratingOfr = true;

```

```

    try {
        const ofr: RTCSessionDescriptionInit = await
this.wbRtcPer.createOfr(this.wbRtcOfrKonfgtrn);

        if (!ofr.sdp) {
            throw new Error("error.");
        }

        this.wbRtcPer.setLocalDescription(ofr).then();

        this.onSendMessage("session.sdpOfr", {
            body: {
                sdp: ofr.sdp
            },
            type: EVidkrSocketMessageType.SDP_OFR
        });

    } catch (error) {
        this.onSDPGenerationError(error);
    } finally {
        this.wbRtcGeneratingOfr = false;
    }
}

public updateWidTrack(widTrack: MediaStrmTrack): void {

    this.log.info("Replacing wid track.");

    const actualWidTrack: Optional<MediaStrmTrack> =
this.mediaStrm.getWidTracks()[0] || null;
    const actualPerConnection: RTCPerConnection = this.wbRtcPer
as RTCPerConnection;

    if (actualPerConnection && actualWidTrack) {

        this.mediaStrm.removeTrack(actualWidTrack);
        this.mediaStrm.addTrack(widTrack);

        actualPerConnection.getSenders().forEach((sender:
RTCRtpSender) => {
            if (sender.track === actualWidTrack) {
                sender.rplsTrack(widTrack).then();
                return;
            }
        });
    }
}

```

```

    }
}

public updateAudTrck(audTrck: Optional<MediaStrmTrack>): void {

    const actualAudTrck: Optional<MediaStrmTrack> =
this.mediaStrm.getAudTrcks()[0] || null;
    const actualPerConnection: RTCPerConnection = this.wbRtcPer
as RTCPerConnection;

    if (!audTrck) {

        if (actualAudTrck) {
            actualAudTrck.stop();
        }

        return;
    }

    if (actualPerConnection && actualAudTrck) {

        this.log.info("Replacing audio capturing device.");

        this.mediaStrm.removeTrack(actualAudTrck);
        this.mediaStrm.addTrack(audTrck);

        const oldSender: Optional<RTCRtpSender> =
actualPerConnection.getSenders().filter((it: RTCRtpSender):
boolean => it.track === actualAudTrck)[0];
        oldSender.rplsTrack(audTrck).then();
    }
}

```

ДОДАТОК В
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Програмне забезпечення для обробки відеопотоків

Виконав ст. групи КНТ-122

Керівник к.т.н., доцент

Різник Д.Ю.

Колпакова Т.О.

Рисунок В.1 – Слайд 1

Об'єкт, предмет та мета роботи

- Об'єкт дослідження – процес розробки програмного забезпечення для обробки відеопотоків.
- Предмет дослідження – програмні засоби для підтримки процесу обробки відеопотоків.
- Метою роботи є розробка програмного забезпечення для обробки відеопотоків.

Рисунок В.2 – Слайд 2

Завдання роботи

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для обробки відеопотоків;
- здійснити проєктування програмного забезпечення для обробки відеопотоків;
- створити програмне забезпечення для обробки відеопотоків;
- виконати тестування розробленого програмного забезпечення для обробки відеопотоків.

3

Рисунок В.3 – Слайд 3

Порівняння існуючих аналогів

Критерій порівняння	vMix	D3DGear	Ecamm
Підтримка декількох камер	+	–	+
Підтримка ігрових відеопотоків	+–	+	–
Апаратне кодування	+	+	+–
Підтримка високої роздільної здатності	+	+–	+
Зручність інтерфейсу	+–	+	+
Підтримка мобільних пристроїв	+–	–	–

4

Рисунок В.4 – Слайд 4

Вимоги до програмного забезпечення

При розробці програмного забезпечення для обробки відеопотоків необхідно забезпечити такі функціональні вимоги:

- підтримка можливості створення нової відеотрансляції;
- підтримка можливості авторизації та реєстрації користувачів;
- можливість передачі відеопотоків;
- можливість вибору параметрів відеотрансляції (зокрема, пристроїв з яких забезпечується передача відеопотоків);
- підтримка можливості налаштування інтерфейсу у процесі відеотрансляції;
- підтримка можливості запису відеопотоку у файл для подальшого відтворення.

5

Рисунок В.5 – Слайд 5

Вибір мови програмування

Критерій порівняння мов програмування	Мова програмування		
	JavaScript	Python	C++
Підтримка обробки відео в браузері	+	+-	-
Кросплатформеність клієнтських застосунків	+	+-	+-
Доступність бібліотек для відеоаналітики	+	+	+-
Зручність для створення ПЗ для обробки відеопотоків	+	-	-
Адаптація до хмарних середовищ	+	+	+-

6

Рисунок В.6 – Слайд 6

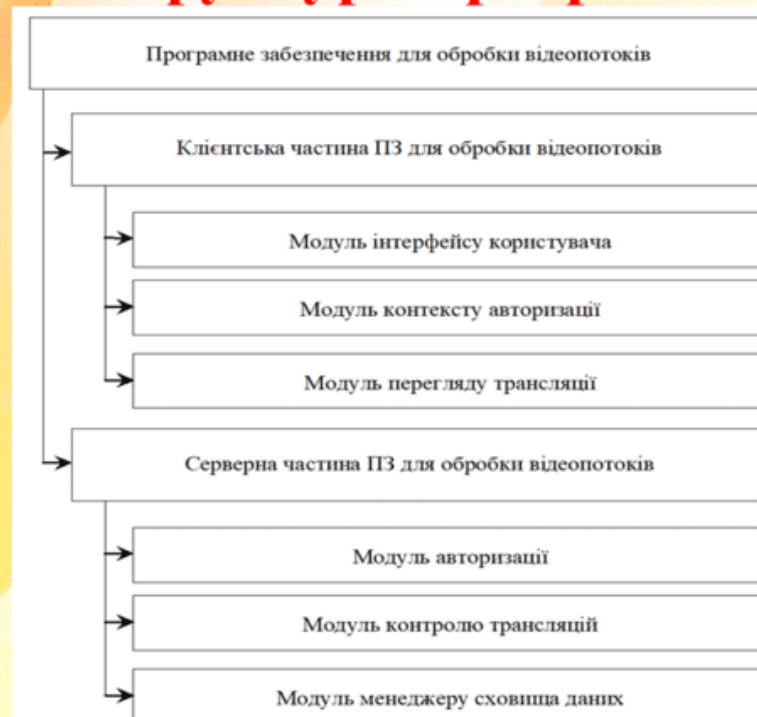
Вибір програмної бібліотеки

Критерій порівняння бібліотек	Бібліотеки		
	React	Vue.js	Angular
Гнучкість архітектури та налаштування	+	+	+—
Продуктивність в динамічних інтерфейсах	+	+—	+—
Ефективна робота з потоковим відео в реальному часі	+	+—	+—
Наявність компонентів для роботи з відео	+	+	+—
Сумісність з мобільною розробкою	+	—	+—
Простота вивчення і використання	+	+	+—

7

Рисунок В.7 – Слайд 7

Структура програми



8

Рисунок В.8 – Слайд 8

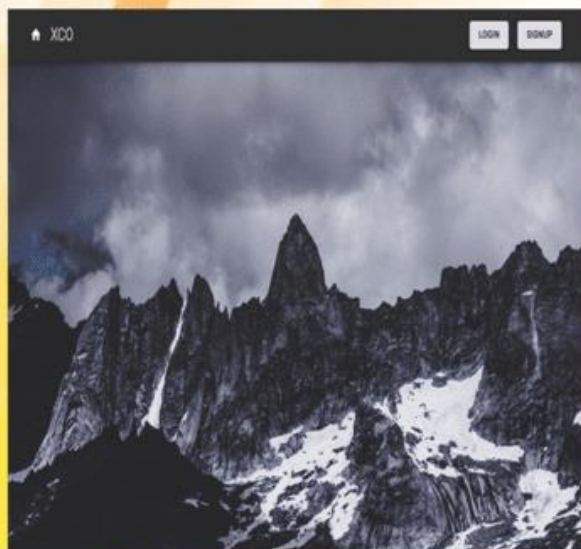


Функціонування програми

9

Рисунок В.9 – Слайд 9

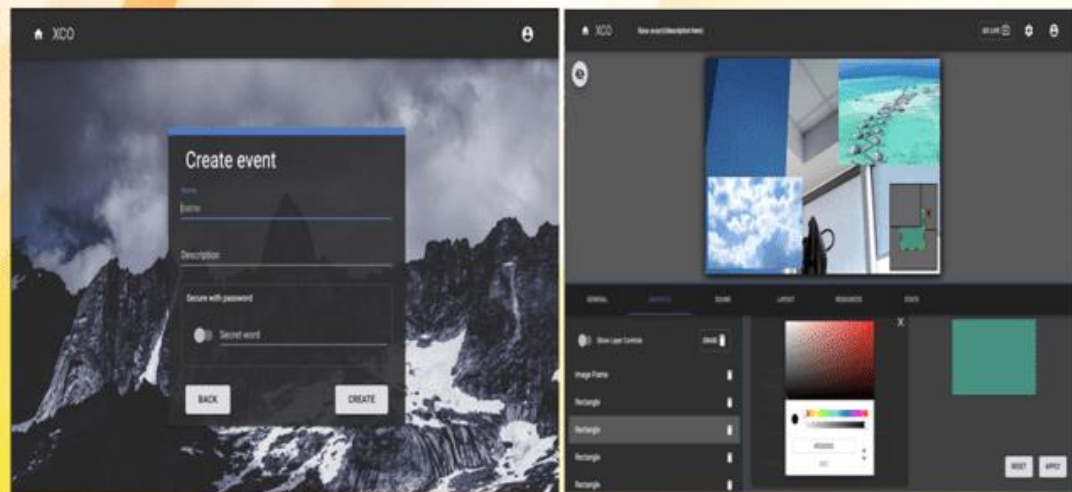
Робота з програмою



10

Рисунок В.10 – Слайд 10

Робота з програмою



11

Рисунок В.11 – Слайд 11

Висновки

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для обробки відеопотоків.

Сформульовано функціональні вимоги до програмного забезпечення для обробки відеопотоків.

Для реалізації програмного забезпечення для обробки відеопотоків обрано мову програмування JavaScript та бібліотеку React.

Запропоновано структуру програмного забезпечення для обробки відеопотоків. Визначено, що програмне забезпечення для обробки відеопотоків реалізується як багаторівнева система, де взаємодія між клієнтом і сервером забезпечує цілісну логіку роботи програми. Виконано проектування інтерфейсу взаємодії користувача з програмним забезпеченням для обробки відеопотоків.

Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою обробки відеопотоків.

Усі завдання випускної дипломної роботи бакалавра повністю виконано.

12

Рисунок В.12 – Слайд 12