Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА

на тему:

Веб-додаток з можливостями запису екранного робочого простору та подальшим редагуванням

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконав: студент 4 курсу, групи 2КІ-18б

спеціальності 123 — Комп'ютерна інженерія

Підцерковний Є.О.

Керівник к.т.н., доц. каф. ОТ

Кожем'яко А. В.

Рецензент дтн, проф. каф. МБІС

Яремчук Ю. Є.

Допущено до захисту

д.т.н., проф. Азаров О.Д.

" 21 " червня 2022 р.

Вінниця ВНТУ — 2022 року

Вінницький національний технічний університет

Факультет _____інформаційних технологій та комп'ютерної інженерії_____

Кафедра _____обчислювальної техніки_____

Освітньо-кваліфікаційний рівень _____бакалавр_____

Спеціальність _____123 — «Комп'ютерна інженерія»_____

ЗАТВЕРДЖУЮ

Завідувач кафедри

обчислювальної техніки

проф. Азарову О.Д.

«07» 02 2022 р.

З А В Д А Н Н Я

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Підцерковному Євгену Олександровичу

1 Тема проекту «Веб-додаток з можливостями запису екранного робочого простору та подальшим редагуванням», керівник роботи к.т.н., доц. каф. ОТ Кожем'яко А. В. затверджені наказом вищого навчального закладу від «24» березня 2022 року №66

2 Строк подання студентом проекту 21.06.2022 р.

3 Вихідні дані до проекту: призначення — записувати екран користувача, та обробляти відео; основні підтримувані функції — запис екрану/вікна/вкладки браузера, редагування відеофайлу.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз предметної області, розробка дизайну рішення, функціональне проектування веб-додатку.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): технічне завдання, стартова сторінка веб-додатку, узагальнена структурна схема.

6 Консультанти розділів роботи приведений в таблиці 1.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Спеціальна частина	Кожем'яко А. В. доцент кафедри ОТ	24.03.22р	17.06.22р

7 Дата видачі завдання «24» березня 2022 р.

8 Календарний план виконання БДР приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів виконання комплексної бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Постановка задачі роботи	08.03.22	вик.
2	Характеристика предметної області	09.03-14.03.22	вик.
3	Аналіз сучасних проектів відеоредакторів та записувачів екрану	17.03-01.04.22	вик.
4	Пошук необхідних технологій для розробки	03.04-09.04.22	вик.
5	Розробка веб-дизайну	12.04-16.04.22	вик.
6	Проектування API веб-додатку	19.04-27.04.22	вик.
7	Розробка функціоналу веб-додатку	01.05-18.05.22	вик.
8	Аналіз виконання роботи, висновки, додатки	21.05-25.05.22	вик.
9	Перевірка якості виконання бакалаврського проекту та усунення недоліків	28.05-09.06.22	вик.

Студент _____ Підцерковний Є. О.

Керівник роботи _____ Кожем'яко А. В.

АНОТАЦІЯ

Підцерковний Є. О. Веб-додаток з можливостями запису екранного робочого простору та подальшим редагуванням. Бакалаврська кваліфікаційна робота зі спеціальності 123 — Комп'ютерна інженерія, Вінниця: ВНТУ, 2022. Загальний обсяг роботи 99 сторінок, 31 рисунок, 2 таблиці.

В даній дипломній роботі було створено веб-додаток, що вміщує в собі 2 веб-додатки — відеоредактор та рекордер екрану, який дозволить користувачам записувати та редагувати відео за допомогою лише веб-браузера, без встановлення будь-якого додаткового програмного забезпечення. Користувач працює з інтерактивним графічним інтерфейсом, щоб виконувати основні функції із запису та редагування відео. По завершенню запису, отримані кадри передаватимуться на сервер для обробки і буде можливість завантаження відео на пристрій. Після завершення редагування створюється список інструкцій, який передається на сервер.

Ключові слова: MLT фреймфорк, HTML, JavaScript, React веб-додаток, відеоредактор, рекордер екран

ABSTRACT

Pidtserkovnyi Ye. O. Web application with the ability to record screen workspace and further editing. Bachelor's degree in specialty 123 — Computer Engineering, Vinnytsia: VNTU, 2022. The total volume of the work is 99 pages, 31 figures, 2 tables.

The aim of this thesis is to create a web application that contains 2 applications — a video editor and a screen recorder, which will allow users to record and edit video using only a web browser, without installing any additional software. The user works with an interactive graphical interface to perform basic video recording and editing functions. Upon completion of the recording, the received frames will be transferred to the server for processing and it will be possible to upload videos to the device. When editing is complete, a list of instructions is created and sent to the server.

Keywords: MLT framefork, HTML, JavaScript, React web application, video editor, screen recorder.

ЗМІСТ

ВСТУП

В теперішній час інтернет об’єднує різні віддалені куточки земного шару. Розвиток і дуже широке поширення інтернету змінює баланс сил між повноцінними десктоп і веб-додатками.

Використовуючи веб-додатки, такі недоліки як недоступність інтернету чи занадто дорогий тарифний план або ж низька пропускна здатність поступово йдуть у минуле. Але в цьому є свої переваги — немає необхідності у установці, налаштуванні, оновленні програмного забезпечення, не використовуються апаратні можливості комп’ютера користувача, і те що дані зберігаються на віддаленому сервері, маючи доступ до них із будь-якої точки планети.

Різноманітні програми для запису екрана полегшили життя багатьом. Записувачі екрана можуть записувати вміст безпосередньо з вашого екрана та дозволяти редагувати, ділитися ним або зберігати його на комп’ютері. Запис екрана — це фантастичний інструмент для тих, хто працює віддалено, для студентів онлайн-класів, геймерів, творців контенту тощо.

Записувачі екрану є одним із найбільш недооцінених інструментів для бізнесу, навіть якщо їх можна легко віднести до найкорисніших основних інструментів, які повинна використовувати кожна організація. Вони можуть бути успішно використані для широкого кола підприємств — більшість із них настільки важливі, що залишається загадкою, чому більшість людей досі відмовляються визнавати їх величезну корисність. Що ще страшніше, так це те, що деякі програми для запису екрана навіть безкоштовні, тому вам навіть не потрібно витрачати гроші на придбання ліцензії.

Але що робить інструмент для запису екрана? У двох словах, це програмне забезпечення, яке дозволяє користувачеві робити цифровий запис вмісту екрана комп’ютера. Від завантаженого робочого столу до відео YouTube, які співробітник таємно дивиться в робочий час, пристрій запису екрана фіксує все, що відбувається на ПК або Mac так, як це бачить користувач. Вони досить актуальні для різних цілей, починаючи від покрокового виконання процедури технічного обслуговування до створення захоплюючих відео-уроків і перевірки

продуктивності ваших співробітників. В поєднанні з іншим програмним забезпеченням, таким як інструменти запису дзвінків, екранні записи навіть використовуються для дистанційного надання психологічної та медичної допомоги.

Зокрема, розвиток мультимедійних технологій вплинув на те, що з кожним днем все більше і більше людей використовують додатки, які дозволяють редагувати відео.

Актуальність теми полягає у розробці додатку, який поєднує в собі 2 субдодатки — відеоредактора та рекордера екрану, що супроводжується потребою у даному продукті в зв'язку з зростанням потреби у медіаконтенті для платформ, наприклад YouTube, а також через збільшення використання у корпоративному секторі дистанційної форми праці

Об'єкт досліджження є технології розробки та методи створення веб-додатків.

Предмет дослідження — створення веб-додатку.

Мета даної роботи є створення веб-застосунку з розширеними функціональними можливостями, який дозволяє записувати екранний простір користувача і також редагувати, обробляти відеофайли. Існує багато як і десктопних, так і веб-додатків, що поступаються доступністю і обмеженим функціоналом, тому в одній із задач є створення додатку, що надавало б більш широкі можливості по запису та редагуванню відео, ніж аналоги.

Для поставлення заданої мети необхідно вирішити такі задачі:

— аналіз предметної області;

— аналіз існуючих програм запису екрану та програм редагування відео;

— вибір технологій проектування для розробки.

— розробка дизайну рішення;

— проектування API для відеоредактора;

— описати структуру проекту;

— функціональна розробка веб-додатку.

1 АНАЛІЗ ТИПІВ ТА ОБЛАСТІ ЗАСТОСУВАННЯ ВІДЕОРЕДАКТОРІВ І РЕКОРДЕРІВ ЕКРАНУ

1.1. Характеристика предметної області

Веб-сервер — комп’ютер, на якому зберігається вся інформація, що знаходиться в мережі Інтернет.[1] На них є встановлено відповідне програмне забезпечення, яке надає можливість користувачам можливість знайти необхідну інформацію. Більша частина такої інформації відображається не екрані користувача у форматі веб-сторінок, які в мережі Інтернет мають своє доменне ім’я або IP-адресу. Аби користувач мав можливість користуватись веб-застосунками з будь-якого девайса чи то смартфон, чи то планшет, чи то ноутбук або ж персональний комп’ютер, то повинна бути встановлений браузер, наприклад, Safari, Google Chrome, Mozilla Firefox, Microsoft Edge, тощо. І за інформацією, яка вводиться в пошуковому рядку — доменне ім’я, браузер буде надавати відповідну сторінку на своєму робочому просторі.[2]

Абсолютно будь-який веб-додаток має в своєму складі одну або декілька пов’язаних між собою веб-сторінок. Причому, кожна сторінка складається із відповідного текстового документа із типом *.html, наприклад, назва замовчуванням для основної сторінки — index.html. В даному файлі описуються спеціальні текстові команди, а саме — код HTML.[3] Він визначає в якому вигляді буде відображатись у вікні браузера. Медіаконтент, а саме: графічні зображення, аудіофайли та відеоматеріали не належать до складу команд у HTML-сторінці, так як вони являються окремими файлами, доступ до яких відбувається через посилання до джерела медіа, які записують у спеціальні команди мови HTML.[4]

Так як веб-додаток веб-застосунок є повноцінним веб-сайтом, то в свою чергу веб-сайт ніяк не можна вважати веб-застосунком.[5] В основному веб-сайти відрізняються від веб-додатків тим, що сайти є лише інформативними, тобто відображає лише інформацію у вікні браузера, а додаток же ж, в свою чергу

має можливість взаємодіяти із користувачем, надаючи йому змогу звертатися до серверів і обмінюватись даними.

На практиці, існують статичні і динамічні веб-сайти.

Статичний веб-сайт — це набір взаємопов'язаних між собою через гіпертекстові посилання HTML-файлів. Дані файли створюються шляхом ручного вводу команд розробником. При їх виклику користувачем, HTML-сторінки повинні бути завчасно завантажені на сервері, після чого будуть відображатись у незмінному вигляді. А як виникне потреба у редагуванні вмісту веб-сторінки, необхідно в ручному форматі видозмінювати текстовий документ із HTML-кодом.[6]

В статичних веб-сайтах є ряд переваг і недоліків. Основними їх перевагами є наступне:

— створення мінімального навантаження на сервер;

— швидке завантаження;

— висока швидкість обробки даних;

— відносно дешева і напрочуд швидка розробка;

— низька складність переносу даних з одного сервера на інший.

Напроти ж, є і недоліки у таких веб-сайтів. Характеризують їх в основному через складність в покращенні працездатності шляхом оновлення сайту і також при модифікації файлів, при закладенні нових функцій та особливостей. Причина криється в тому, шо без знань в сфері веб-програмування керування такими сайтами звичайному користувачу буде даватись якщо не з великими складнощами, то взагалі буде не в стані що-небудь змінити у ньому. [7]

Динамічні сайти мають широкий асортимент функцій, наприклад, у них є можливість редагувати інформацію, публікувати щоденні новини або вести ваш блог без знань у сфері веб-програмування, але якщо основною діяльністю користувача на цьому веб-ресурсі буде поглинання інформації, то такий ресурс все одно буде вважається веб-сайтом, а не веб-додатком.[8]

Динамічні сайти, мають куди більш гнучкі налаштування, ніж статичні. Їх переваги криються в легкості розробки «з нуля» шляхом ручного внесення всіх

скриптів і кодів розмітки, баз даних, тощо. Зокрема, частіше можна помітити, що динамічні сайти побудовані на спеціальних системах керування контентом — CMS. Дані системи містять в собі «прямо з коробки» уже готові скрипти і модулі, які можна використати при розробці.[9] Якщо ми візьмемо за основу CMS, то можна буде розробити незчислену кількість динамічних сайтів. На сьогоднішній день найбільшою кількістю користуються наступні CMS — WooCommerce, OpenCart, Magento, PrestaShop, Joomla.

Ще одна перевага динамічних веб-сайтів, яка є основною, полягає в їх інтерактивності, тобто користувач становиться активним учасником, коли він шукає певну інформацію, коли натискає на кнопки інтерфейсу, коли він постійно вводить текст, бо просто тривалий час працює з клавіатурою та мишкою. Дуже влучним прикладом веб-додатка є дуже популярний на теперішній час сайт tiktok.com, де користувач не лише поглинає контент, а і майже постійно проводить певні дії як: поставити вподобайку, поширення відео, зберегти собі аби потім переглянути, записати і розмістити нове відео, і також спілкування в чаті із друзями.

1.2 Аналіз програм для редагування відео

При розробці відеоредактора, є досить важливим дослідити уже існуючі аналоги. Було підібрано редактори, які користуються популярністю у відомих авторів відеохостингу YouTube, на основі рекомендацій моїх знайомих.

Існуючі аналоги можна поділити на 2 категорії — десктопні відеоредактори та онлайн відеоредактори. Огляд існуючих рішень зосереджений в основному на проектах, які можуть використовуватись безкоштовно або мають безкоштовну ліцензію.

1.2.1 Десктопні відеоредактори

Серед десктопних відеоредакторів є ряд безкоштовних продуктів, які надають безкоштовну ліцензію для своїх користувачів, і є придатні для індивідуального та комерційного використання. Загалом, інтерфейс даних

додатків схожий, обширний, при чому користувачеві відразу надається можливість застосування частини функцій для простого редагування — обрізати, скоротити, переміщати, компонувати відеоряди. Їх недоліками являється висока потреба в часі, так як повільно вчишся працювати із ними та мають високі вимоги до апаратної частини, тобто аби працювати із комфортом постає необхідність у досить потужному персональному комп'ютері або ноутбуку.[10]

Рішення, які мають відкритий код мають в своєму складі проекти Shotcut і Kdenlive. Shotcut — мультиплатформний відеоредактор з відкритим кодом. Kdenlive — безкоштовний редактор для KDE4. Ці два проекти використовують фреймворк MLT для обробки мультимедійних файлів. Тому навіть у безкоштовних програмах можна створити будь-який проект, а не лише в рішеннях, де застосовуються методи платних ліцензій, причому використовуючи перший варіант, користувач не є обмежений ніяким чином.[11] Під час тестування безкоштовних аналогів, які мають за основу безперервне збереження проекту були відзначені проблеми із стабільністю. Інтерфейс обох програм є майже ідентичним і зображено на рис. 1.1.

Верхній рядок меню містить глобальні функції проекту, такі як відкриття проекту, збереження, створення, перевірка та відтворення проекту. Ліва колонка використовується для керування даними проекту, керування фільтрами та переходами вибраного обраного елемента на часовій шкалі, налаштування властивостей і параметрів медіаоб'єктів. Зокрема, обидві програми використовують вкладки і закладки для перемикання вигляду. Правий стовпець показує вікно попереднього перегляду отриманого відеоряду. Аби керувати областю попереднього перегляду, застосовуються елементи плеєра, що розміщені під відео.

Монтажний стіл розташований нижче. Базові функції для редагування відео знаходяться над лінією часу. Поточний відрізок часу візуалізовано вертикальною лінією. Сама лінія часу характеризується будь-якою кількістю аудіо- та відеодоріжок, причому аудіодоріжки позначаються зеленим кольором, а відеодоріжки — синім.

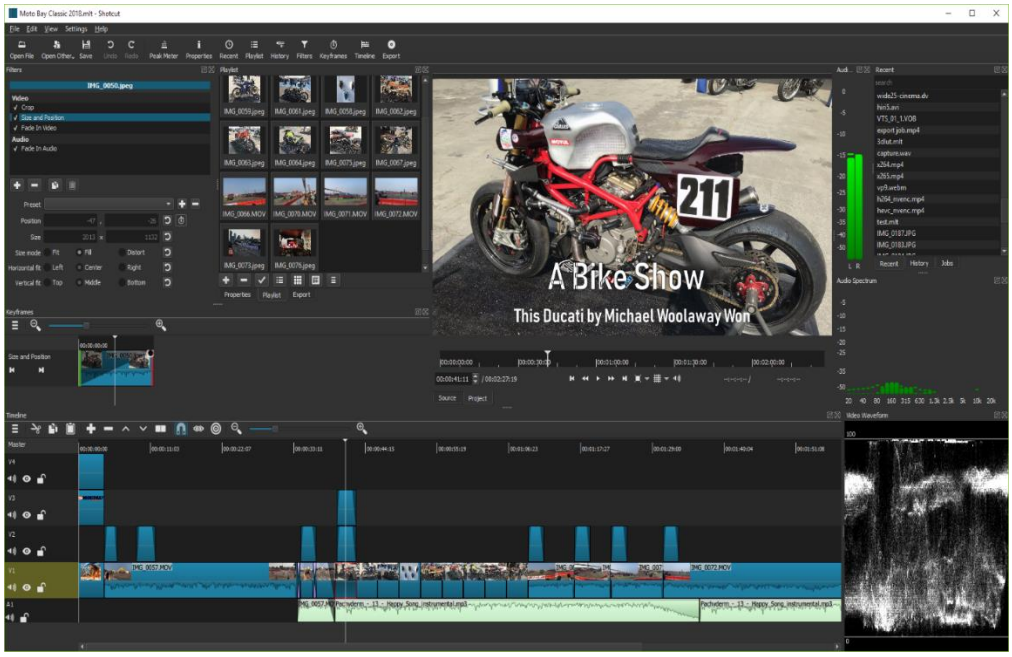

Рисунок 1.1 — Інтерфейс користувача нелінійного відеоредактора Shotcut з виділеним розташуванням елементів управління.

Безкоштовною програмою, яка не використовує фреймворк MLT, є, наприклад, програма Openhot, яка перейшла на власне ядро. Його інтерфейс дуже спрощений порівняно з попередніми двома редакторами. Макет залишається незмінним, лише обмежуючи кількість інформації, яку вона надає користувачеві. Угорі розташовані ті ж дії для проекту, ліворуч бібліотека файлів, фільтрів і переходів, праворуч — попередній перегляд, а внизу — часові шкали та налаштування властивостей фільтра/переходу. Незважаючи на спрощення інтерфейсу, тут не бракує жодної функції.

Високоякісні фірмові програми включають iMovie від Apple, Adobe Premiere Pro від Adobe Systems і VEGAS Pro від Sony. Професійні програми відрізняються від безкоштовних в основному більшою стабільністю, більш складними фільтрами та шаблонами. Ліцензії на Adobe Premiere Pro та Sony Vegas Pro коштують близько 620 гривень/місяць та 770 гривень/місяць, так як вони пропонують свої продукти лише у формі підписки, але Sony надає можливість придбати за повною ціною — 15500 гривень. З цих програм я використав iMovie. Програма від Apple ділить робочу область на такі ж групи, рис. 1.2.

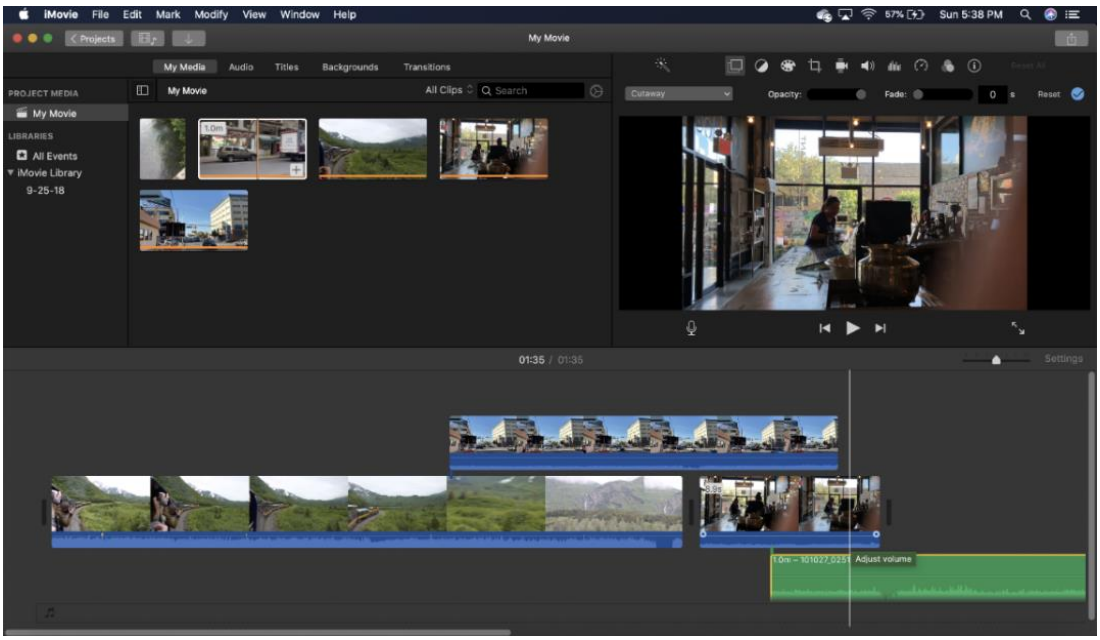

Рисунок 1.2 — Інтерфейс користувача iMovie з виділеним макетом керування

На відміну від попередніх програм, він відображає в лівій половині не тільки файли поточного проекту, а й файли в бібліотеках. Пропоноване рішення також має стосуватися спільної бібліотеки із загальними ресурсами, якими користуються, наприклад, користувачі факультету. Застосування фільтрів різне. Фільтри знаходяться над попереднім переглядом відео. При натисканні на піктограму з’являється меню фільтрів. Користувач ніколи не встановлює числові значення. Налаштування фільтрів вирішують візуальні повзунки, крапельниці та інші інструменти. Цікавим рішенням є лінія часу, яка не має лінійки з часом. Шкала часу змінюється на лінії. Між кліпами є пробіл, хоча кінець попереднього та початок наступного кліпу є одночасно. З моєї точки зору, це тривожний елемент, під час відтворення індикатор поточного часу має властивість перескакувати між відеорядами і змінює свою швидкість. З іншого боку, переходи можна зручно вставляти в дані пробіли. Елементи на часовій шкалі не можна переміщувати, можна змінити лише їх порядок. Якщо нам потрібно мати порожнє місце у відео, ми повинні вставити однорідний колір з бібліотеки (вкладка «Фон»), що є не зручно. Панель інструментів відсутня над часовою шкалою. Інструменти для елементів шкали часу доступні через контекстне меню правою кнопкою миші або за допомогою комбінацій клавіш. Відео доріжки

налаштовані на синій колір, аудіо доріжки на зелений. Колірна гамма цього проекту заснована на програмі iMovie.

Цікавим проектом серед настільних програм є редактор з відкритим кодом fwf — відеоредактор з використанням бібліотеки FFmpeg. Програма написана на JavaScript-фреймворку Vue.js і скомпільована за допомогою інструменту Electron. Відеоредактор має всі базові функції, він не дуже надійний, але він може бути натхненний технологіями, які використовуються для реалізації інтерфейсу користувача, рис. 1.3. Програма не підтримує переходи. Він використовує бібліотеку vis.js на часовій шкалі. vis.js — JavaScript бібліотека для візуалізації дат. Хронологія — це лише один із інструментів візуалізації бібліотеки. Інструмент використовується для відображення подій епохи Unix (00:00:00,000 1 січня 1970 року), тоді як проект починається в момент часу 0. Програма використовує шрифт Source Sans Pro, опублікований під безкоштовною ліцензією Open Font License та є відгалуженням використаного шрифту Material Icons від Google (ліцензія Apache-2.0).

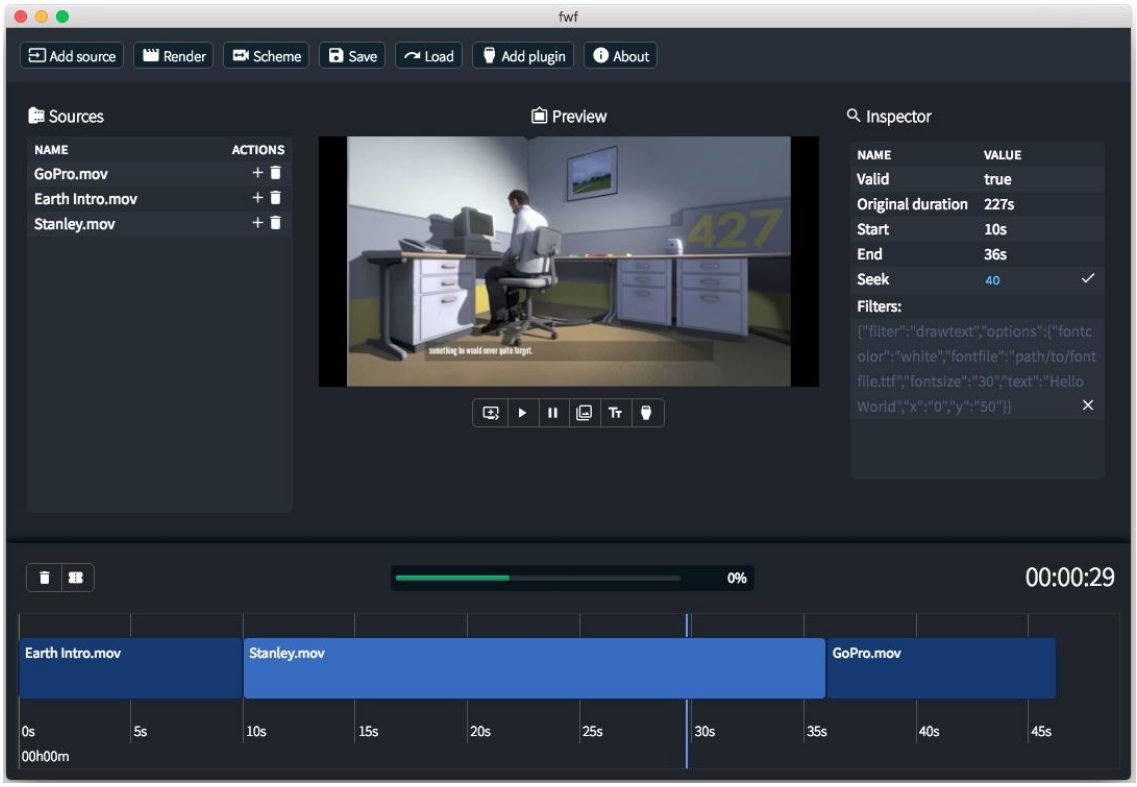

Рисунок 1.3 — Інтерфейс користувача редактора fwf, складений інструментом Electron

1.2.2 Веб-відеоредактори

Для веб-редакторів існує велика різниця між платними і неоплачуваними рішеннями. Платні програми мають багато функцій, безкоштовні рішення мають серйозні обмеження. Я не знайшов жодних рішень з відкритим кодом і безкоштовними ліцензіями.

Платні рішення з функціями та інтерфейсами ближче до рідних програм. Серед платних редакторів згадую WeVideo, Magisto, Loopster, Animatron і Clipchamp. WeVideo представлено в п’яти цінових варіантах. Найдорожчий варіант не обмежує розмір файлу або максимальну довжину. Інтерфейс користувача дозволяє працювати з відеофайлами під час завантаження на сервер. Попередній перегляд відтворюється в браузері, отримане відео на сервері. Після завершення візуалізації користувачеві надсилається електронний лист. Відео, створене у безкоштовній версії, містить водяний знак.

Magisto пропонує також платні варіанти. Найдешевший обмежує максимальну тривалість результуючого відео 2,5 хвилини. Найдорожче обмежує 10 хвилин. Часова шкала недоступна під час створення проекту. Відображає кліпи як послідовність елементів, подібно до файлових менеджерів. Фільтри та переходи не можна встановлювати для елементів окремо, вони встановлюються глобально для всього проекту за допомогою шаблонів. Безкоштовний варіант пропонує просте 3-етапне редагування відео з обмеженими функціями.

Loopster пропонує вихідну роздільну здатність 1080p у всіх планах. Основною проблемою для обробки лекцій є максимальна тривалість — 30 хвилин для найдорожчого варіанту.

Animatron в основному використовується для створення коротких трейлерів і реклами. Це дозволяє створити проект максимальною тривалістю 10 хвилин у найдорожчому варіанті. Безкоштовний план не дозволяє завантажувати відео, а лише розміщувати відео в соціальних мережах. Під час роботи з редактором я помітив проблему з різною орієнтацією одного й того самого відео в редакторі.

Для невимогливих користувачів може бути достатньо інструменту Clipchamp Create, який також пропонує безкоштовний варіант з роздільною здатністю відео, обмеженою 480p. Розмір проекту та максимальна довжина не обмежені, відео не має рекламного водяного знака. У найдорожчому варіанті він дозволяє створювати відео з роздільною здатністю до 1080p. Інтерфейс ідентичний настільним програмам, рис. 1.4. Редактор працює лише в Google Chrome, а відео обробляється на стороні веб-переглядача. Користувач не повинен залишати відеоредактор під час обробки. Час обробки відео до 2 хвилин займає близько 20 хвилин, навіть у роздільній здатності 480p. На додаток до завантажених файлів, він надає бібліотеку з аудіо та відео.

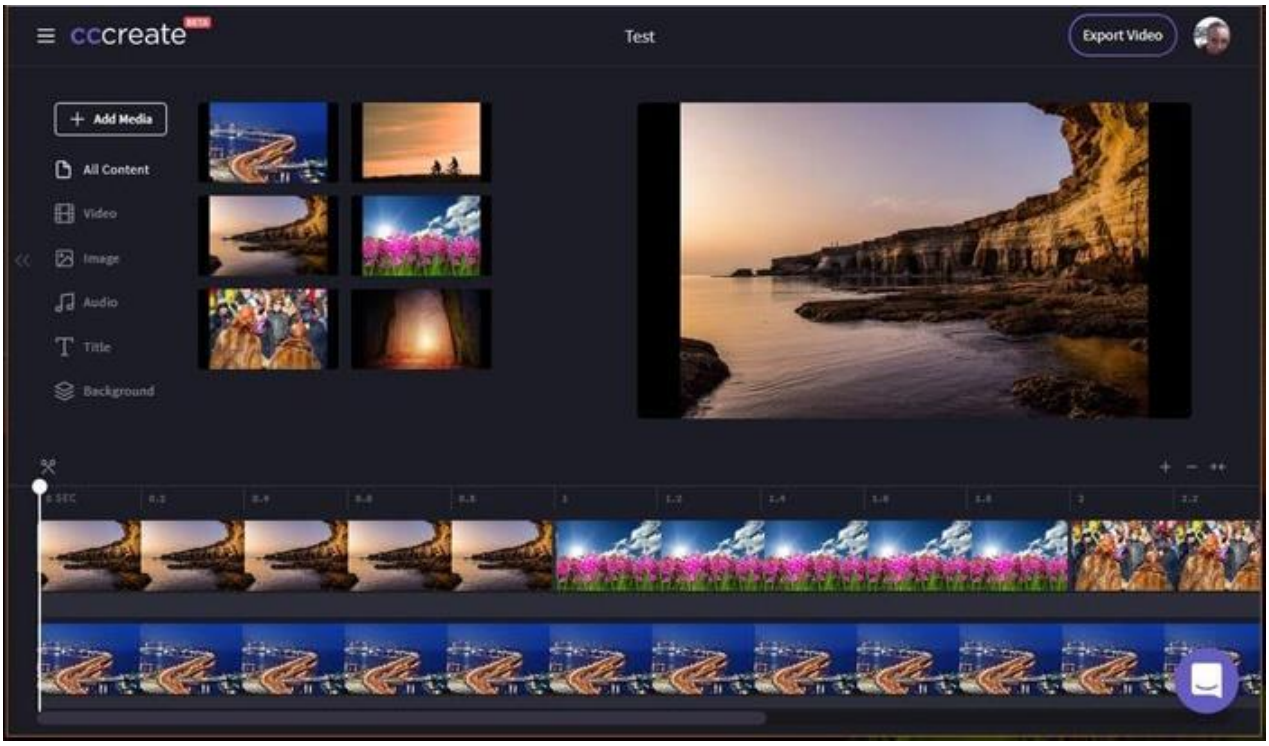

Рисунок 1.4 — Інтерфейс користувача онлайн-лінійного відеоредактора Clipchamp Create

Для повноти згадаю два інших редактора, жоден із них не є задовільним — Kizoa та Movie Maker Online. Редактор Kizoa створений в Adobe Flash, містить достатню кількість ефектів, переходів і налаштувань, рис. 1.5. Окрім завантажених файлів, він надає власну бібліотеку зображень, відео та аудіо. Він показує одну часову шкалу внизу, це лінійний редактор. Попередній перегляд

отриманого відео відображається у спливаючому вікні, під час відтворення попереднього перегляду виникає проблема зі стабільністю роботи браузера. Основним недоліком є посилання на Adobe Flash Player, який сьогодні в занепаді і закінчиться в 2020 році.[13] Безкоштовний варіант пропонує 2 хвилини відео 720p, найдорожчий варіант відео 4K без обмежень довжини.

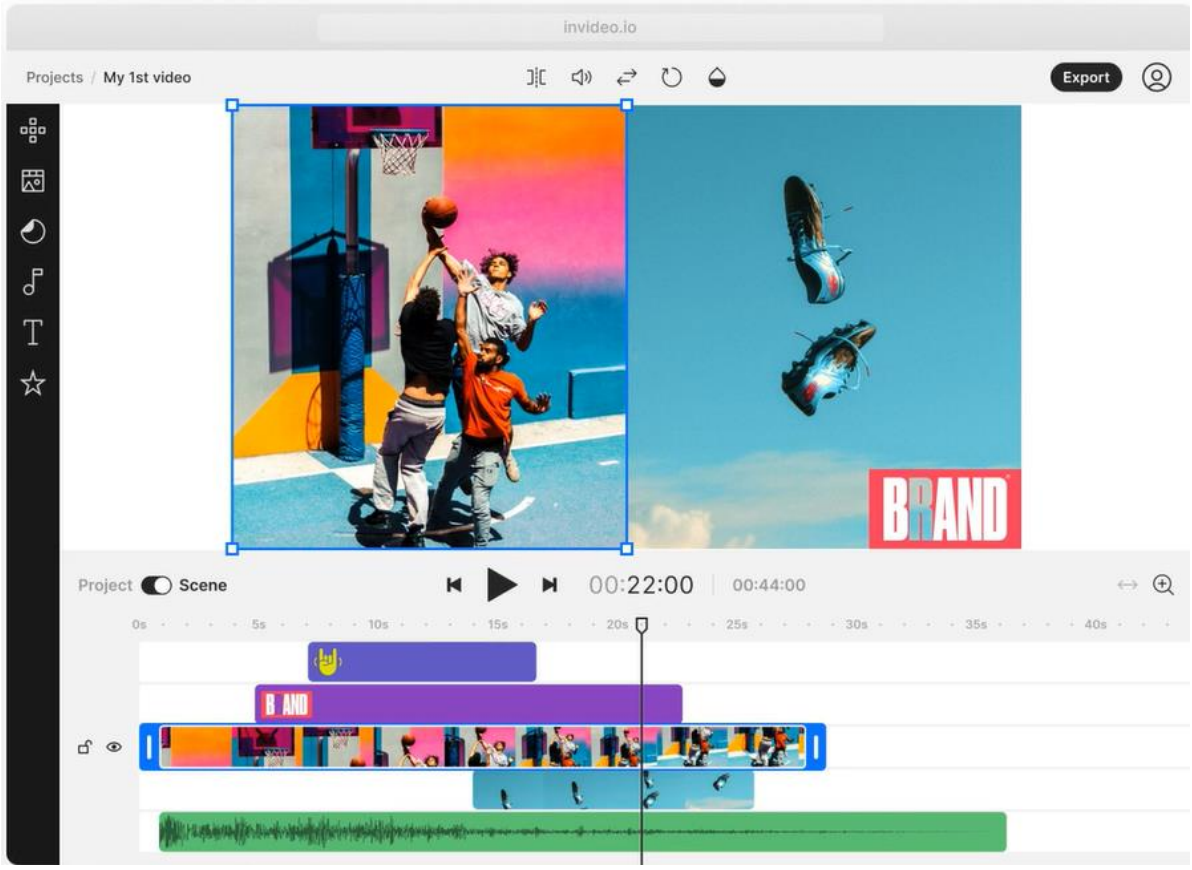

Рисунок 1.5 — Інтерфейс користувача онлайн-лінійного відеоредактора Kizoa

Movie Maker Online представляє відеоредактор з нетрадиційним інтерфейсом користувача, рис. 1.6. Він не використовує плагіни і відображає вісь відео вертикально. Це нелінійний редактор, який має 4 доріжки — аудіо, фонову доріжку, основну доріжку та доріжку накладання. Додати або видалити трек неможливо. Редактор оформлений як одна сторінка, на якій окремі кроки розташовані один під одним. Файли завантажуються на першому кроці, потім користувач повинен перейти до кроку 2 і відредагувати. Він містить обмежену кількість фільтрів, і переходи можна змішувати за допомогою двох доріжок з поступовим збільшенням/зменшенням. Для користувачів настільних додатків це

зовсім інший інтерфейс, в якому буде важко орієнтуватися. На відміну від попереднього рішення, Movie Maker Online не дозволяє зберігати або експортувати проект і використовувати його пізніше. Програмою можна користуватися без реєстрації, отримане відео з роздільною здатністю 720p забезпечено водяним знаком і не повинно перевищувати 10 хвилин. Програма має запропонувати підписку, але наразі вона недоступна.

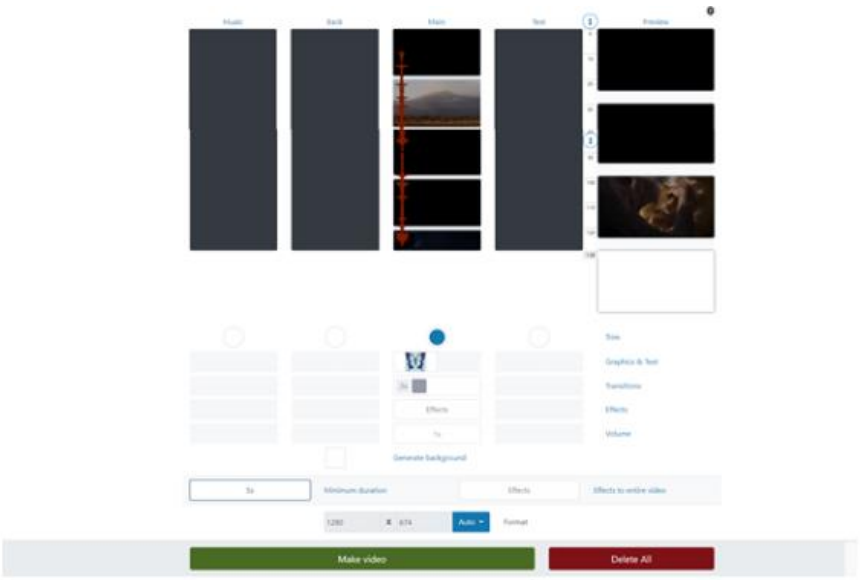

Рисунок 1.6 — Інтерфейс користувача онлайн-відеоредактора Movie Maker Online

1.3 Аналіз програм для запису екрану

Доволі довгий час записувати дії на екрані було справжньою проблемою. Аби це зробити, потрібно було використовувати складні інструменти або платити експерту з даного програмного забезпечення, що не завжди було можливим. На щастя, зручність програмного забезпечення для запису екрана значно покращилася з роками. Вже сьогодні є доступним можливість створювати професійні підручники з мінімальними зусиллями. Окрім навчальних посібників, є кілька інших причин, чому можна захотіти записати свій екран. Якщо проводити будь-які демонстрації продуктів, потреба чи пропозиція у технічній підтримці чи використання відеодзвінків, використання екрана за замовчуванням на комп’ютері буде недостатньо хорошим для зйомки

високоякісних кадрів. Але якщо є бажання отримати більш точний спосіб запису всього екрана, або бажання зафіксувати певну частину, високоякісний записувач екрану буде чудово підходити для здійсненн цієї мети.

Програмне забезпечення для запису екрана може записувати весь (або частину) екрана вашого комп'ютера або мобільного телефону. Запис може включати все, від ваших натискань і рухів курсора до введення URL-адреси у ваш браузер, щоб допомогти людям дізнатися, що і як робити.[12] Доступна аудіо-розповідь, а іноді також пропонуються анотації як частина пакета інструментів. Однак немає двох однакових інструментів для запису екрана. Деякі інструменти для запису екрана не можуть зберігати записані кадри в різних форматах або експортувати їх безпосередньо на популярні відеоплатформи, як-от YouTube .

Інші можуть мати можливість робити все це, але їм може не вистачати однієї чи двох функцій. Наприклад, не кожна програма для запису екрана постачається з записом екрана в HD якості. Тому слід звернути увагу на його можливості, перш ніж завантажувати або встановлювати програмне забезпечення для запису екрана на свій пристрій. Також важливо оцінити власні потреби у записі. Можливо, не потрібно записувати в HD. Можливо, нещодавно придбали відеохостинг і можна обійтися без можливості експорту на YouTube. Найкращою програмою записом екрана для вас буде той, який відповідає вашим конкретним потребам.

Функціональність в автономному режимі є ключовою відмінністю між десктопними і веб-рішеннями. У той час як пристрої для запису екрана для настільних комп'ютерів можуть працювати без активного з'єднання, онлайн-рекордери вимагають стабільного Інтернету.

1.3.1 Десктопні рішення

OBS Studio є найкращим безкоштовним записом екрана для захоплених геймерів. На відміну від надзвичайно популярного FRAPS (який дозволяє записувати лише 30 секунд, якщо ви не купили ліцензію, і застосовує водяний

знак до отриманого матеріалу), OBS Studio є відкритим вихідним кодом і повністю безкоштовним для використання без обмежень.

OBS Studio підтримує як потокове передавання, так і запис у високій чіткості без обмежень щодо кількості чи тривалості творінь. Є можливість транслювати в прямому ефірі на Twitch або YouTube ігри, зберігати проекти та повертатися до них пізніше, або кодувати свої кадри у форматі FLV і зберігати їх локально.

Оскільки OBS Studio може записувати використовуючи відеокарту, вона може записувати ігри, що працюють у повноекранному режимі (багато інших пристроїв запису екрана можуть записувати, лише якщо гра працює у вікні), за допомогою настроюваних гарячих клавіш для керування записом. OBS Studio також може повністю використовувати багатоядерні процесори для підвищення продуктивності та записувати зі швидкістю 60 кадрів в секунду (або навіть вище). Може знадобитися деякий час, щоб налаштувати його саме так, як ви хочете, але OBS Studio є, безумовно, найкращим і найпотужнішим записом екрана для геймерів. Інтерфейс програми наведений на рис. 1.7.

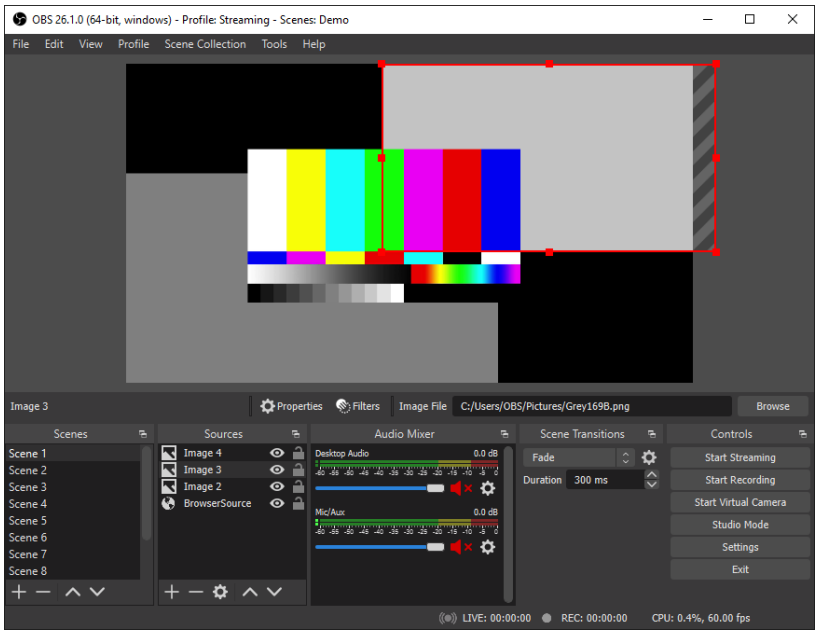

Рисунок 1.7 — Інтерфейс користувача відеорекордера OBS Studio

FlashBack Express є безкоштовною версією платної програми, він не буде накладати потворні водяні знаки на ваші записи або обмежувати час, а також

наповнений функціями та інструментами, які конкурують з багатьма програмами преміум-класу. Його інтерфейс менш лякає, ніж OBS Studio, і показаний на рис. 1.8.

Наявна можливість записувати на весь екран, вікно, вибрану область або веб-камеру. По завершенню, ваш запис з’явиться в простому редакторі, де можна обрізати його відповідно до ваших потреб, а потім експортувати на YouTube, на FTP-сервер або на ПК. Це досить стандартна комплектація як для безкоштовної програми для запису екрана, але в розширених налаштуваннях є безліч продуманих налаштувань, які зроблять записи справді професійними. Диктофон може автоматично приховувати паролі , введені на екрані, замінювати дурні шпалери на звичайні, приховувати брудні значки на робочому столі та виділяти вказівник миші, щоб було легше стежити за ним. Також є спеціальний ігровий режим, який дозволяє визначити кількість кадрів, записаних за секунду. Тривалість записів не обмежена. Можна розділити довгі записи на шматки — блискучий штрих, який допоможе уникнути створення величезних, громіздких файлів. Записи також не будуть мати водяні знаки.

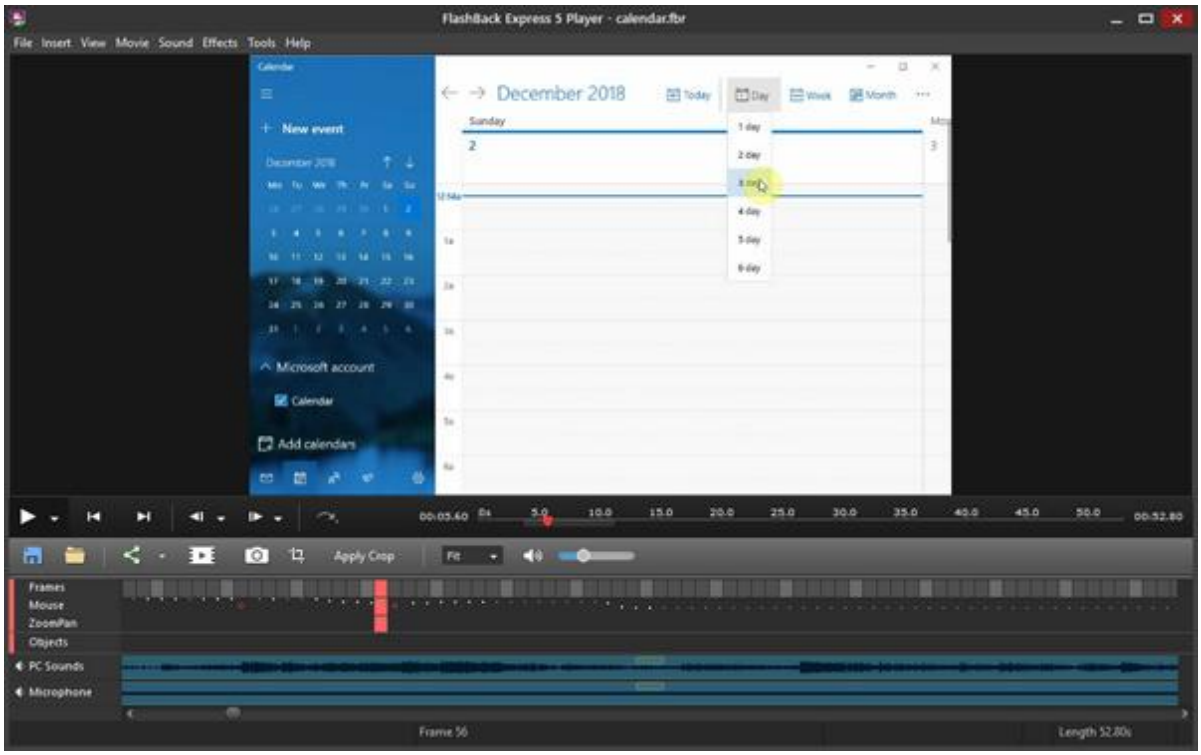

Рисунок 1.8 — Інтерфейс користувача відеорекордера Flashback Express

Після останнього оновлення Flashback Express може експортувати у формати WMV, AVI та MP4, усуваючи потребу в окремому відеоконвертері, і він більше не потребує безкоштовного ліцензійного ключа для активації.

Розробник Blueberry Software також випустив безкоштовний запис екрану, розроблений спеціально для ігор — FBX. Цей рекордер записує в грі HD з апаратним прискоренням для карт AMD і Nvidia, а також процесорів Intel.

Apowersoft Unlimited — набір програмних додатків, що включає програми запису екрана для Android, iOS, Mac і Windows. Це дозволяє бездротово транслювати пристрій Android або iPhone на екран комп'ютера та використовувати настільний диктофон, щоб одночасно записувати аудіо з ПК, мобільний пристрій, мікрофон і веб-камеру комп'ютера. Запис екрана на робочому столі пропонує кілька режимів запису, включаючи повноекранний режим, спеціальну область тощо, а також дозволяє користувачам коментувати запис у режимі реального часу. Ви можете застосовувати виноски, рядки, текст тощо, не роблячи паузи.

За допомогою його інтуїтивно зрозумілих програм для мобільних пристроїв вам просто потрібно налаштувати свій комп'ютер і смартфон під одну мережу Wi-Fi і почати трансляцію. Інтерфейс програми наведений на рис. 1.9.

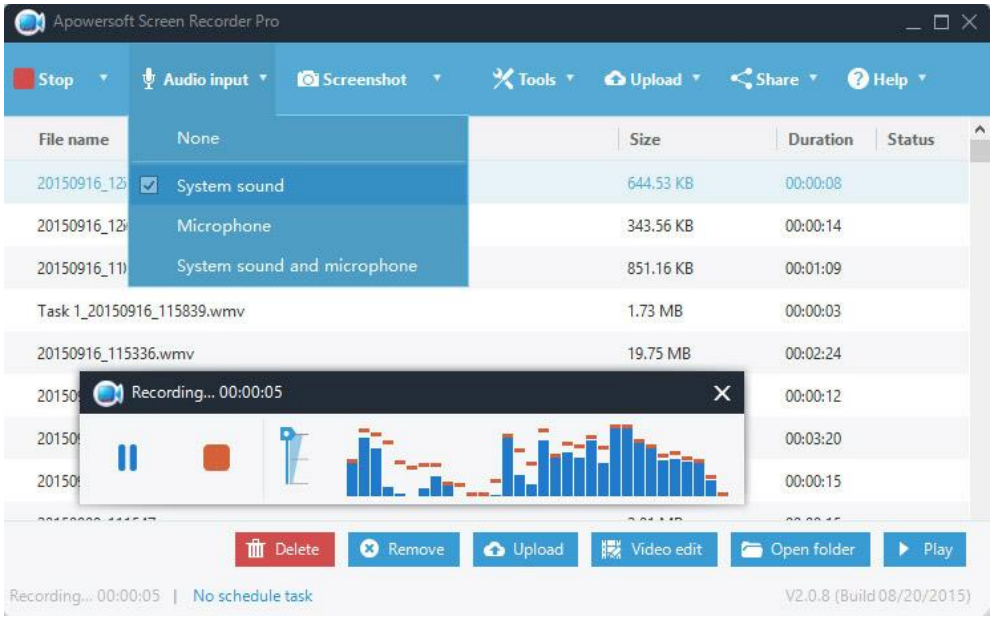

Рисунок 1.9 — Інтерфейс користувача відеорекордера Apowersoft Unlimited

Користувачі Mac і Windows можуть безпосередньо використовувати пристрій для запису екрана на робочому столі, щоб записувати всі види діяльності на екрані.

ShareX — програма з відкритим вихідним кодом для запису фотографій екрана та відео. Немає водяних знаків або обмежень за часом, але інтерфейс не є найінтуїтивнішим у світі, тому найкраще використовувати комбінації клавіш. Для швидкої довідки ви можете розпочати запис, натиснувши Shift + Print Screen, і зупинити його знову за допомогою Ctrl + Shift + PrintScreen. Можна використовувати цей безкоштовний запис екрана, щоб зберегти знімок як GIF, а не відеофайл, що може бути надзвичайно корисним для обміну на форумах і в соціальних мережах.

ShareX не тільки є одним з найкращих безкоштовних пристроїв для запису екрана, але він також може захоплювати всю веб-сторінку, що прокручується, захоплювати та ідентифікувати текст за допомогою OCR і навіть знімати ваш екран відповідно до сценарію. Інтерфейс програми наведений на рис. 1.10.

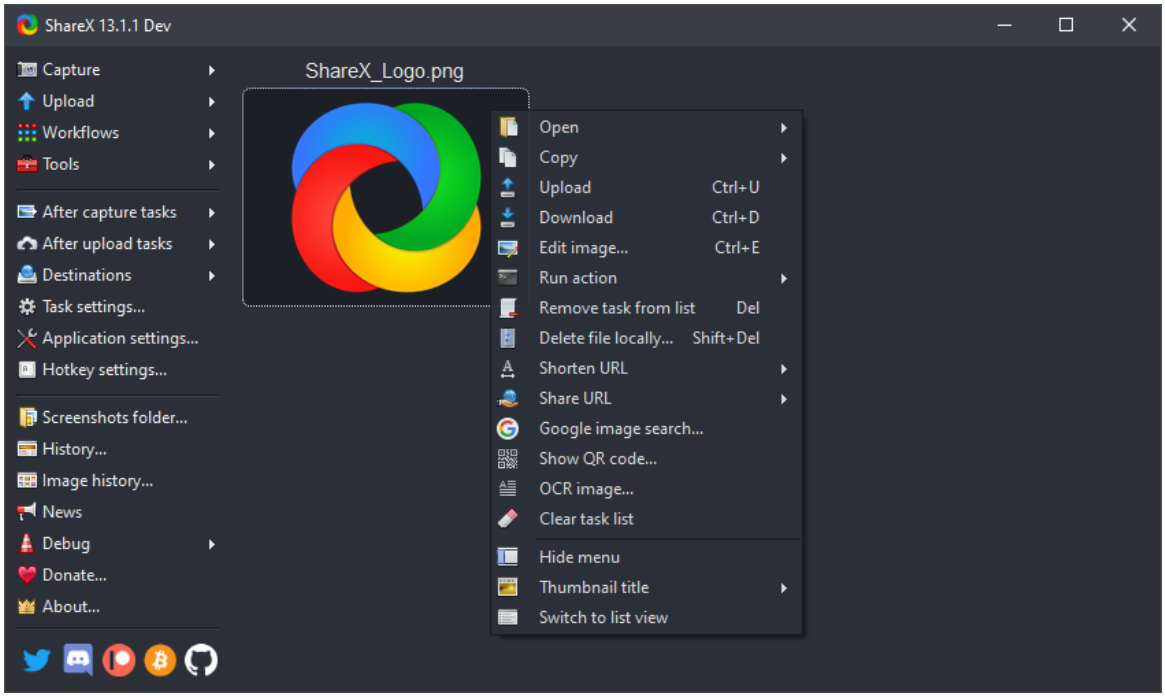

Рисунок 1.10 — Інтерфейс користувача відеорекордера ShareX

Ще одна з його найкращих функцій — це можливість надсилати зняті захоплення та відео прямо на сайт для обміну файлами або в соціальні мережі.

Існує величезна кількість варіантів, з якими ви можете ознайомитися на сайті проекту. На жаль, ShareX не підтримує знімки екрана або записи з ігор, які працюють у повноекранному режимі.

Bandicam — це надійна програма для запису екрана, яка підтримує запис екрану та ігрового процесу. Інтерфейс програми показаний на рис. 1.11. Він використовує високий коефіцієнт стиснення без шкоди якості відео. Bandicam дозволяє знімати відео 4K надвисокої чіткості і в 120 кадрів в секунду. Ще однією важливою особливістю є можливість запису пристрою Bandicam. Отримавши окрему карту захоплення, ви можете записувати відео з IPTV, HDTV, Apple TV, смартфона, PlayStation і Xbox. Крім того, Bandicam дозволяє користувачам робити знімки екрана у форматах JPEG, PNG і BMP.

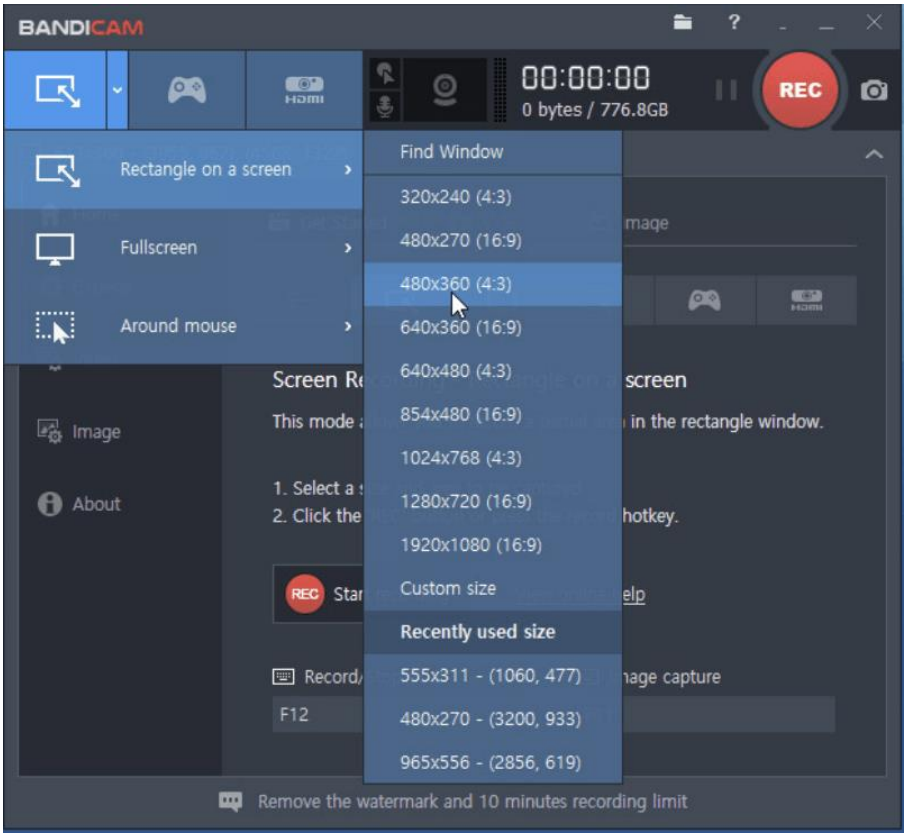

Рисунок 1.11 — Інтерфейс користувача відеорекордера Bandicam

По суті, можна використовувати Bandicam для запису майже всього, включаючи весь екран комп'ютера, електронні таблиці Excel, веб-браузер, презентації Powerpoint тощо. Його безкоштовна версія розміщує водяний знак на

відео, тому потрібно стати платним користувачем, щоб повністю використати його потенціал.

1.3.2 Рішення на веб-архітектурі

Для користувачів, які хочуть чогось простого, зручного та зрозумілого: онлайн-рекордери — найпростіші у використанні. Можна використовувати їх прямо зі свого браузера, для завантаження потрібно лише розширення Chrome або веб-сайт. Ці рекордери дуже залежні від Інтернет-з'єднання, особливо для завантаження готового запису. Онлайн-рекордери не ресурсоємкі, тому вам не потрібен потужний комп'ютер, і є можливість зберігати свої записані файли в Інтернеті. Тому розглянемо декілька аналогів такого типу рекордерів. Інтерфейс користувача наведено на рис. 1.12.

VEED з'явився із надзвичайно потужним записувачем екрана, що є сьогодні, хоча це всього лише розширення для браузера.

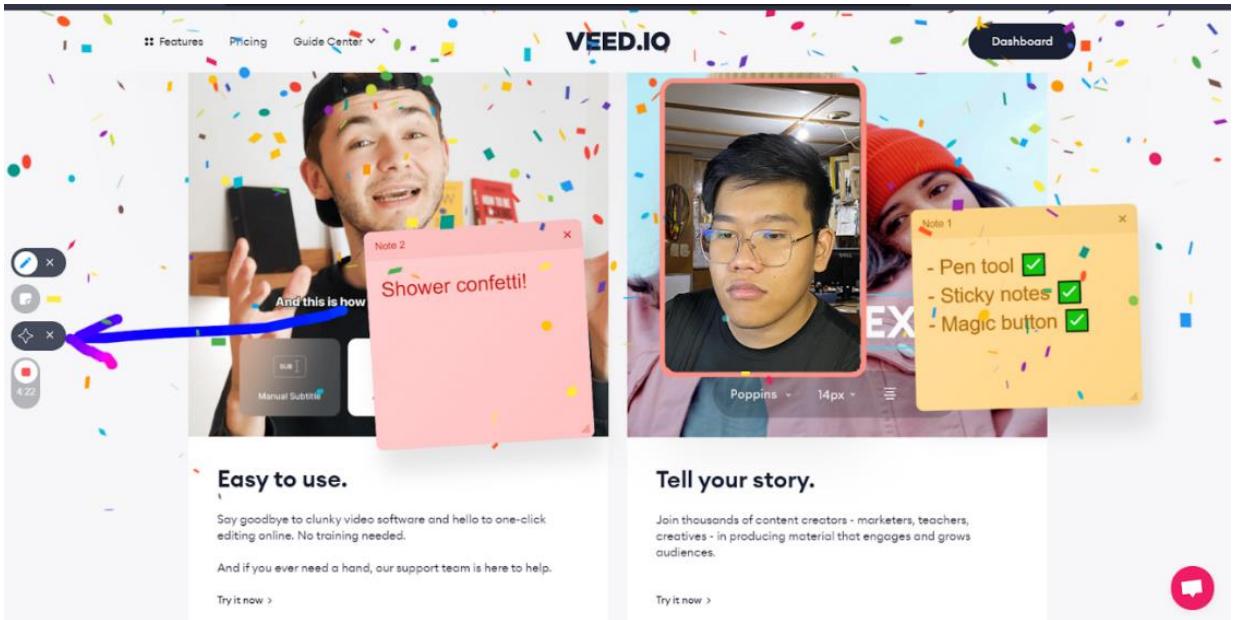

Рисунок 1.12 — Інтерфейс користувача онлайн-відеорекордера VEED

Рекордер VEED — це новий інструмент, але він уже попереду з точки зору функцій. Ви можете записувати необмежену кількість відео, скільки забажаєте. Крім того, користувачі також мають можливість записувати свої мікрофони та веб-камери одночасно.

Для тих хто є креативним дизайнером, є також можливість малювати поверх свого відео. Функція липких нотаток — це чудовий спосіб відстежувати все, що пояснюється. Цей інструмент ідеально підходить для викладачів і творців контенту, яким часто потрібно записувати свій екран. На щастя, навіть початківець може легко використовувати рекордер VEED. Інтерфейс користувача приведено на рис. 1.13.

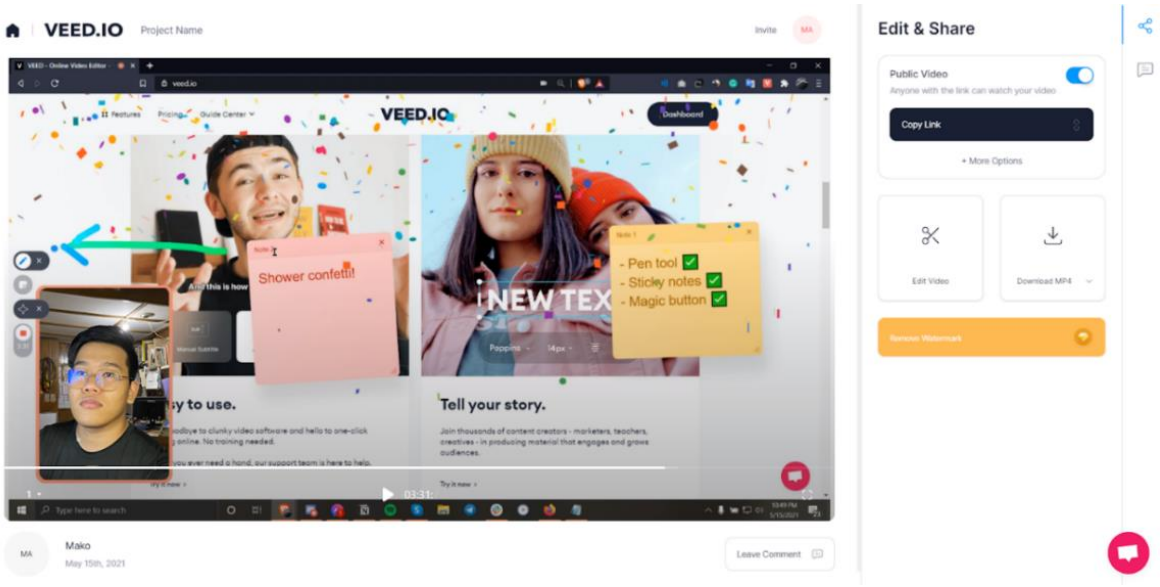

Рисунок 1.13 — Інтерфейс користувача онлайн-відеорекордера VEED

Для корпоративних цілей, одним із найвідоміших інструментів для запису екрана — Loom. Інтерфейс рекордера наведено на рис. 1.14.

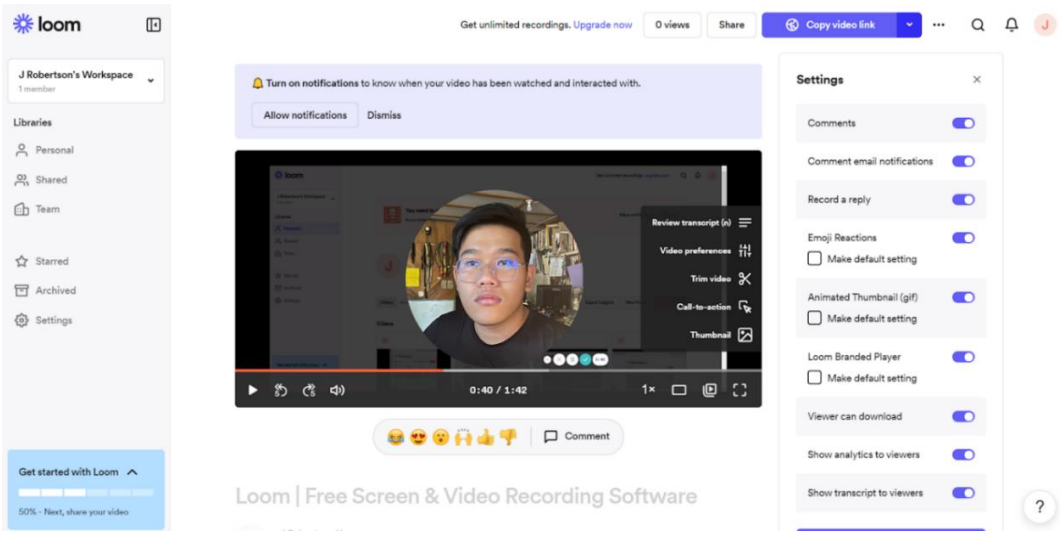

Рисунок 1.14 — Інтерфейс користувача онлайн-відеорекордера Loom

На перший погляд, Loom — це простий екранний записувач. У ньому можна вибирати між кількома макетами запису на екрані, і він також записує вашу веб-камеру. Loom не є найкращим з точки зору функціоналу. На відміну від VEED, неможливо додавати нотатки під час запису. Функція малювання також обмежена платним доступом, що є неприємністю. З іншого боку, можна використовувати Loom через розширення Chrome. Крім цього, можна налаштувати розмір веб-камери під час запису. Це чудовий спосіб переключити увагу глядача з вмісту на обличчя, особливо коли потрібно щось пояснити за допомогою веб-камери.

Також він має недоліки — він дозволяє лише 5 хвилин використати для запису, досить тривалий процес реєстрації. У ньому також є обмеження на 100 відео та скріншотів для безкоштовних облікових записів, але цього більш ніж достатньо для більшості людей. Одне, що Loom робить правильно — це зручно ділитися своїм відео. Готовий результат автоматично завантажується, коли ви закінчите, і ви можете просто надіслати посилання своїм колегам, щоб отримати доступ до відео.

2 ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ РІШЕННЯ

Основною метою даної роботи є створення веб-додатку, що вміщує в собі відеоредактор і рекордер екрану, який буде працювати у всіх сучасних браузерах, а обробка відео відбуватиметься зі сторони сервера. Даний додаток повинен виконувати всі завдання за допомогою JavaScript та асинхронних запитів, не перезавантажуючи всю сторінку. Через навантаженість інтерфейсу користувача у відеоредакторі не рекомендується вибирати чистий JavaScript. Варто використовувати саме фреймворк JavaScript, аби полегшувати операції і допомагати створювати чистий код.

Найбільш використовувані фреймворки: React, Backbone.js, RequireJS, AngularJS або Vue.js (використовується програмою fwf).[14] З цих фреймворків я вибрав React, який є водночас і шаблонною системою і має якісну документацію.

Веб-додатку також знадобиться серверна частина, на яку клієнтська частина завантажуватиме файли, описуватиме серверу файлові та кадрові операції та інструктуватиме отримане відео. Ця архітектура, з точки зору мережі, називається клієнт-серверна. З погляду дизайну, для відеоредактора буде використовуватись архітектура MVC (Model-View-Controller). Також необхідно забезпечити зв'язок між клієнтом та сервером. Для обміну даними через мережу обидві частини повинні використовувати певний формат серіалізації. JavaScript найчастіше використовує XML або JSON. JSON (JavaScript Object Notation) є новішою та запис JSON є дійсним кодом запису об’єктів в JavaScript. З іншого боку, XML (Extensible Markup Language) дозволяє передавати двійкові дані. Код JavaScript обмінюватиметься даними із сервером, тому краще вибрати формат JSON.

Для кожної частини додатку буде свій сервер, що має 2 завдання. Спочатку потрібно буде надати користувачеві файли HTML, CSS і JavaScript, а потім відповісти на асинхронні запити JavaScript. Під час проектування я вибирав між реалізацією сервера за допомогою PHP та із використанням Node.js. PHP виник

у часи статичних сторінок та особистих веб-сайтів, Node.js зазвичай асоціюється з односторінковими програмами. Я вибрав Node.js саме тому, що планую створити відеоредактор і рекодер як односторінкові програми, через можливість використовувати одну мову для клієнтської та серверної частини, а також через асинхронний доступ до блокуючих операцій. Навпаки, я використовував би PHP при створенні інтернет-магазину, блогу або ж інформаційної системи.

Сервер відеоредактора клієнту буде доступний через інтерфейс (API), коли у рекордері сервер буде доступний тільки через сокет. Інтерфейс буде використовуватися для виклику дій на серверах та для отримання даних. Існує кілька концепцій зв'язку та створення API, таких як XML-RPC, SOAP або REST. XML-RPC і SOAP використовують XML, як формат серіалізації для обміну даними, але через JavaScript, що є на стороні клієнта і сервера, я вибрав JSON. REST - це спосіб зв'язку, що базується на HTTP-запитах. REST не прив'язаний до HTTP, але немає практичної причини використовувати інший протокол. Формат серіалізації за замовчуванням для обміну даними — JSON, але можна використовувати, наприклад, і XML.

Також необхідно обробляти мультимедійні файли. Однак, ні веб-браузер, ні веб-технології не призначені для візуалізації відео; рендеринг відео у браузері неефективний. Редактор призначено для обробки відео на сервері. Для цього можна використати FFmpeg для рекордера і MLT — для відеоредактора. Наприклад, FFmpeg використовує редактор fwf або Android Video Editor, який використовує FFmpeg1. FFmpeg використовується через командний рядок. Кожна операція являє собою одну команду (почати запис, закінчити запис, об’єднати кадри). Дуже залежить від порядку операцій, що для рекордера екрану буде достатньо. Проект описуватиметься файлом із послідовністю команд для FFmpeg або однією довгою командою. Мультимедійне середовище MLT також пропонує свої інструменти через командний рядок. Проект може бути описаний послідовністю команд, а також із використанням формату серіалізації XML. Весь проект описується сутностями XML, і файл XML потім передається лише у XML файл. XML-файл постійно зберігатиметься на сервері, і сервер редагуватиме

його відповідно до вимог клієнта. REST API не має стану, тобто запит має бути виконаний, а статус проекту повинен зберігатись у XML-файлі проекту. У проекті не використовуватиметься система баз даних.

Перелік використовуваних технологій, їх розташування та зв’язок наведено на рис. 2.1.

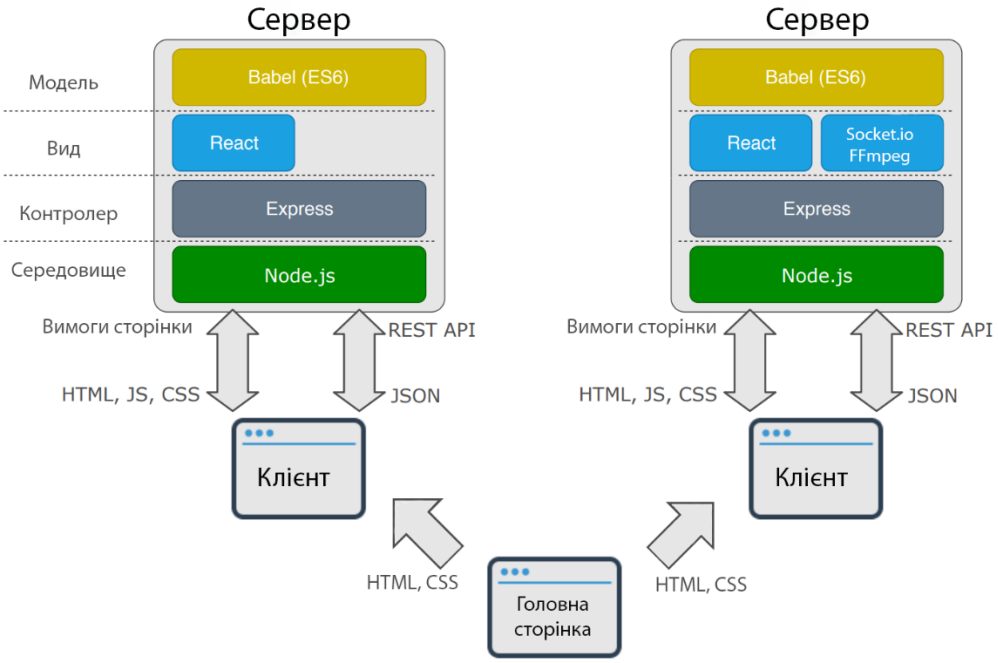

Рисунок 2.1 — Використовувані технології у додатку з точки зору архітектури клієнт-сервер та з точки зору дизайну архітектури MVC

2.1 Вибір технологій проектування для розробки

У даному підрозділі узагальнено використані технології, що були показані на схематичному рис. 2.1. У ньому згадуються подібні технології, пояснюються відмінності, підсумовуються переваги проекту та описується використання, які розглянемо нижче.

Node.js створює середовище виконання JavaScript, і побудоване на движку JavaScript V8 від Google. Він дозволяє використовувати JavaScript як системну мову, яка має доступ до пам'яті, буферів, процесів, файлів і сокетів.

Мета Node.js — забезпечити простий спосіб створення масштабованих мережевих додатків. Node.js не використовує потоки, заснований на подіях і

асинхронне виконання операцій блокування, при цьому забезпечуючи легкою модульністю. Високої продуктивності можна досягти, усунувши операції блокування, використовуючи асинхронну обробку, що керується подіями. Node.js використовує один потік для всіх запитів і використовує псевдопаралельну обробку замість паралельної обробки.

Для порівняння швидкості виконання програми наведемо приклад простої програми, в якій лічильник інкрементується в мільярд разів в порожньому циклі. Час виконання програми для кожної мови є таким: JavaScript (Node.js v9.3.0) 0,635 секунди, мова C (Apple LLVM v8.1.0) 2,398 секунди і Python (v3.6.1) 31,096 секунди. Усі тести проходили на ноутбуці з процесором Intel Pentium 2,3 ГГц. Движок V8 компілює JavaScript замість інтерпретовування, тому можна використовувати мову високого рівня, не відмовляючись від швидкості компіляції мов.[15]

JavaScript на стороні клієнта є обов'язковим для інтерактивного веб-додатка. Завдяки JavaScript, що працює на клієнтській і серверній частинах, програмісти можуть використовувати в проекті одну мову. Це також відкриває можливість використання одного і того ж коду на сервері та на стороні клієнта. Все, що вам потрібно зробити, це правильно розділити функції на модулі, а потім імпортувати ці модулі як на стороні сервера, так і на стороні клієнта.

Це є мова програмування з синтаксисом, натхненним мовою C, яка перекладена на JavaScript. TypeScript є розширенням JavaScript, програма в JavaScript є дійсною програмою в TypeScript. Він забезпечує потужний контроль типів, класи (сьогодні та в ES6), інтерфейс, успадкування.

Автономний переклад JavaScript виявляє синтаксичні помилки та, за бажанням, забезпечує сумісність зі стандартами ES3 і ES5 (альтернативою є Babel і ESLint2 для перевірки коду на наявність синтаксичних помилок). Він також додає багато «синтаксичного цукру» - гетери і сетери, типи перерахування, проміси. Він підтримує інтеграцію як з Angular 2, так і з React.[16]

З іншого боку, перевірка типів є повною, лише якщо всі пакети містять файли заголовку TypeScript. TypeScript необхідно включити в процес розробки за допомогою Node.js і React. Під час розробки, JavaScript необхідно компілювати щоразу, коли змінюється вихідний код. Для програмістів-початківців і для невеликих проектів це створює складнощі, які можуть не переважити переваги впровадження.

З появою, стандарт ES6 перейняв від TypeScript оголошення змінних, дійсних у даному фреймворку без помітних ефектів (let), можливість створення класів, успадкування, решти параметрів, промісів тощо. Головною перевагою залишається сильна типізація та перевірка синтаксису під час компіляції.

Через мій попередній досвід роботи з PHP, мені не потрібні були ці функції, тому я вирішив використовувати стандарт ES6 і пізніші версії. Я забезпечив зворотну сумісність клієнтського JavaScript інструментом Babel. Я виконав синтаксичну перевірку за допомогою інструмента ESLint.

Щоб полегшити створення веб-сервера, я використав Express. Express — це веб-фреймворк для Node.js. Він надає бібліотеки, корисні для забезпечення основних функцій веб-сервера (маршрутизація, підтримка системи шаблонів для генерації сторінок, обробки запитів, параметрів, сеансів і файлів cookie, аутентифікації, генерування відповідей, обробки помилок тощо).

React — це бібліотека JavaScript для створення інтерфейсів користувача. Він заснований на компонентах, станах компонентів і сигналах між компонентами. Бібліотека дозволяє розділити інтерфейс користувача на ізольовані частини коду - компоненти. Кожен компонент має свій власний стан, свої власні функції і має повний контроль над його відтворенням. Взаємозв’язок компонентів зазвичай ієрархічна, збираючи компоненти, ми можемо створити як і простий, так і складний інтерфейс, зберігаючи чистий код.

Стан програми зберігається окремими компонентами, стан не має одного центрального сховища. Іноді ця функція може не підійти. У цьому випадку просто створюється кореневий компонент і вставляються вихідні компоненти в кореневий компонент. Тоді сховище стану може бути кореневим компонентом.

У лістингу 2.1 показано, як може виглядати компонент React.

Лістинг 2.1 — Приклад реалізації компоненту React

```
import React, { Component } from 'react';
import SourcesTableRow from './SourcesTableRow';
import Uploader from './Uploader';
export default class SourcesTableRow extends Component {

render() {
return (
<tr>
<td>
<div><i className="material-icons resource-preview" aria-hidden=
        "true">panorama</i></div>
</td>
<td>
{this.item.name}<br/>
{this.item.duration !== null && <small>Délka: {this.item.duration}</small>}
</td>
<td className="column-right">
<button onClick={() => this.props.onInsert(this.item.id)}><i
className="material-icons" aria-hidden="true">control_point</i></button>
<button onClick={() => this.props.onRemove(this.item.id)}><i
className="material-icons" aria-hidden="true">delete</i></button>
</td>
</tr>
);
}}
…
```

На прикладі лістингу 2.2 показано ієрархію компонентів. Актуальний компонент — Sources, компонент посилається на SourcesTableRow і Uploader у тілі функції візуалізації. Компонент Sources буде головним над цими двома компонентами. Якщо компонент Sources є кореневим, на нього ніхто посилається. Потім потрібно вказати, де повинна відображатися сторінка HTML. Ми досягаємо цього у файлі JavaScript, який Webpack компілює в клієнтський файл JavaScript.

Лістинг 2.2 — Приклад реалізації візуалізаці сторінки через React

```
ReactDOM.render(<Sources items = {{}} onAddResource={()=>{}}/>,
document.getElementById('app'));
```

Коли користувач відвідує сторінку з компонентами React, він спочатку отримує HTML-файл із скелетом сторінки. Під HTML-кодом буде посилання на файл JavaScript, який завантажить, виконає та замінить елементи скелета, що відображаються компонентами React. У цьому випадку вміст елемента замінюється відображуваним компонентом Sources, що вказано у лістингу 2.3.

Лістинг 2.3 — Розмітка сторінки у компоненті Sources

```
<!DOCTYPE html>
<html lang=’ua’ dir=”ltr”>
<head>
<title> React page</title>
</head>
<body>
<div id=”app”>
<h2>Завантажено</h2>
Якщо програма не завантажується, перевірте, чи ввімкнено JavaScript
</div>
<script src=”./main.js” async></script>
</body>
</html>
```

Якщо код повинен бути зрозумілим, варто створити компоненти як клас, який розширює клас Component. Рекомендується використовувати окремий файл для кожного класу та вказати клас із компонентом в якості експорту за замовчуванням. Це дозволяє нам використовувати інші компоненти, імпортуючи файл.

Клас компонента може містити власні функції, а також функції, які мають особливе значення. Функція конструктора класу є загальною. На початку конструктора ми повинні викликати конструктор предку. Конструктор зазвичай встановлює стан компонента, атрибути класу, а також фіксує контекст для змінної this всередині функцій.

Іншими важливими методами, які мають особливе значення, є: componentDidMount, componentDidUpdate, componentWillUnmount та render.

Метод render відтворює компонент. Суть функції полягає у поверненні JSX-коду компонента. JSX — це розширення синтаксису JavaScript. За своїм

синтаксисом JSX більше нагадує XML, ніж HTML. Усі теги об’єднані в пари (повинні бути закриті). Код повинен бути вкладений в кореневий елемент. Якщо ми не хочемо відображати кореневий елемент, ми можемо використовувати порожні теги <></>. JSX поєднує HTML і JavaScript. Команди та змінні JavaScript доступні в дужках — {this.props.items}. Атрибути елементів HTML записуються у CamelCase (з великої літери). Атрибути мають ті ж значення, що й змінні JavaScript. Особливим є атрибут style, який приймає об’єкт із властивостями CSS.

Решта методів викликаються під час вставки компонента у віртуальне дерево DOM, яке створює React, під час оновлення та при видаленні з DOM.

Щоб інтерфейс користувача працював, компоненти повинні взаємодіяти один з одним. Компоненти мають деревоподібну структуру, комунікація відбувається у двох напрямках — від предка до нащадка і навпаки. Компоненти можуть передавати параметри дочірнім компонентам у міру їх створення.

Ми можемо передати зворотні виклики або статус компонента. У дочірньому компоненті ці параметри доступні через об’єкт this.props. Якщо ми передаємо стан дочірньому компоненту, а дочірній компонент працює безпосередньо зі станом у функції візуалізації, стан предка також оновиться і перемалює дочірній компонент. Наприклад, якщо ми змінимо стан кореневого компонента, ця зміна може «бульбашкувати» до компонентів нижчими на декілька рівнів, і все відбувається автоматично. Зв'язок із коренем здійснюється за допомогою зворотних викликів. Дочірній компонент може змінити стан батьківського лише дотично. Він викликає функцію, яку батьківський компонент зробив доступною йому в this.props, і на основі виклику батьківський компонент змінює свій стан і обидва компоненти перемальовуються.

Стан компонента встановлюється методом this.setState. Як параметр передається об’єкт з елементами, які потрібно перезаписати. Попередній і наступний стан не повинні змінюватися. Якщо ми хочемо відредагувати лише один елемент об’єкта, ми повинні створити копію об’єкта та встановити цю копію як новий стан.[17]

Являє собою JavaScript-бібліотеку для мобільних та веб-додатків, яка реалізує обмін даними в реальному часі. Також, як і Node.js, Socket.io є подіє-орієнтованою. Встановлюється за допомогою менеджера пакетів.

Socket.io використовує протокол WebSocket, але при необхідності, може використовувати і інші методи, наприклад, Adobe Flash, JSONP або AJAX запити, надаючи скрізь однаковий інтерфейс. Крім того, вона має багато інших функцій, в список яких входить трансляція на декілька сокетів, збереження даних і асинхронний ввід/вивід.

Реалізується бібліотека по транспортному протоколі у реальному часі. Однак, через узгодження протоколу у клієнта, що підтримує стандартний WebSocket, можливості звязатись із сервером Socket.io нема. Також, як і клієнт, що реалізує цю технологію, не може зв’язуватись із сервером WebSocket на основі NonSocket.io чи Long Polling Comet. Тому Socket.io вимагає використання своїх бібліотек як на стороні клієнта, так і на стороні сервера.

FFmpeg — найвідоміший набір бібліотек, опублікованих під вільною ліцензією, який дозволяє записувати, декодувати та передавати, мультиплексувати аудіо- та відеопотоки.

Даний набір бібліотек працює на багатьох архітектурах (зокрема, x86, amd64 та arm) та під поширеними операційними системами (GNU/Linux, Windows, MacOS, Android).

Для розробки був обраний FFmpeg як активний. Крім того, FFmpeg непогано документований.

Компоненти FFmpeg:

— libavutil — допоміжна бібліотека зі стандартними загальними підпрограмами для різних компонентів FFmpeg;

— libavcodec — бібліотека для роботи з аудіо та відеокодеками, причому більшість кодеків були розроблені для з нуля для забезпечення найкращої продуктивності;

— libavformat — бібліотека з мультиплексорами та демультиплексорами для різних аудіо- та відеоформатів;

— libavdevice — бібліотека для читання та запису відео- та аудіопотоків з пристрою;

— libavfilter — бібліотека, яка дозволяє змінювати відеопотік між декодером та кодером "на льоту";

— libswscale — бібліотека масштабування відео;

— libswresample (libavresample) — бібліотека для передискретизації аудіопотоків.

FFmpeg підтримує велику кількість кодеків і дає можливість їхньої гнучкої настройки. FFmpeg надає мінімальний шар абстракції, тому її можна назвати низькорівневою бібліотекою, що є перевагою та недоліком одночасно. З одного боку, такий підхід дає змогу гнучко працювати з кодеками, з іншого — платою за це є складність розробки. FFmpeg поширюється під ліцензією LGPL 2.1, деякі частини — під GPL. Завдяки ліцензії LGPL можна використовувати бібліотеку в мобільних клієнтах для попереднього перегляду слайдшоу. Крім того, бібліотека FFmpeg добре документована та має активну спільноту користувачів, тому вона і була обрана для розробки бібліотеки.

2.2 Дизайн інтерфейсу користувача

Цільова група — комп’ютерно грамотні користувачі, які іноді користувалися програмою для редагування відео і програмою для запису екрану. Інтерфейс вибрано так, щоб він і виглядав сучасно, просто, і також нагадував класичні настільні програми, допомагаючи користувачам орієнтуватися. Типовим завданням для рекордера екрану було б записати робочий простір якої-небудь програми, а для відеоредактора було б редагувати лекцію в навчальний відеоролик або створити навчальний посібник із запису з камери або екрана. Додаток буде використовуватися на комп’ютерах і ноутбуках, адаптувати інтерфейс під екрани мобільних пристроїв немає необхідності через громіздкий інтерфейс відеоредактора. Управління буде здійснюватися за допомогою клавіатури та миші, сенсорні екрани не очікуються, але не виключаються.

Першочергово, по завантаженню головної сторінки у користувача буде можливість вибору яким із додатків користуватись. Для цього надаються 2 кнопки на вибір. Інтерфейс головної сторінки наведено на рис. 2.2.

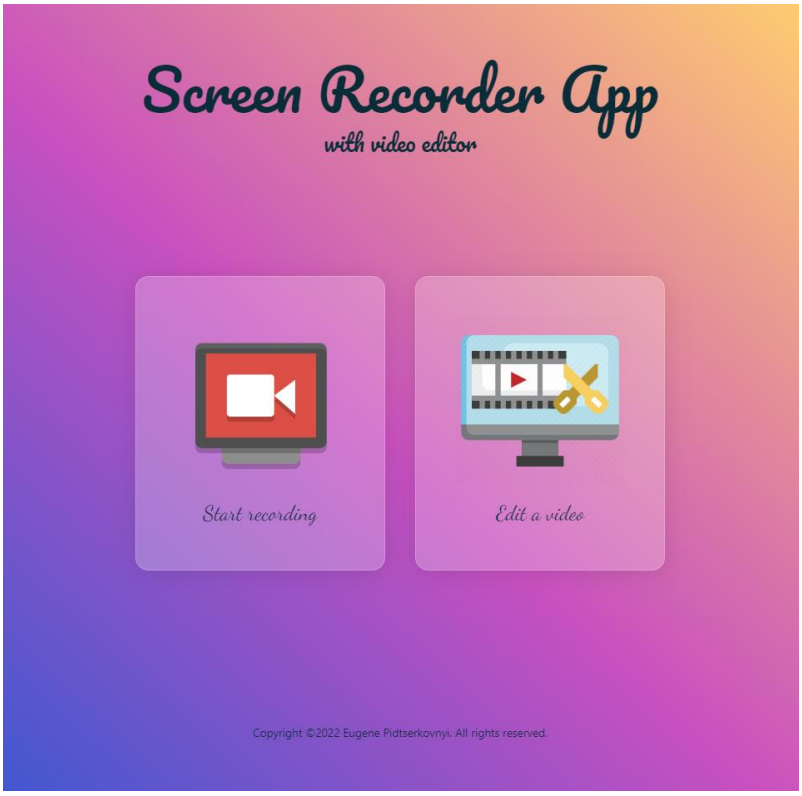

Рисунок 2.2 — Користувацький інтерфейс головної сторінки

Після завантаження початкової сторінки програми для запису екрана відображається сторінка що складається з 2 частин — сам заголовок і кнопка для початку запису, що показано на рис. 2.3.

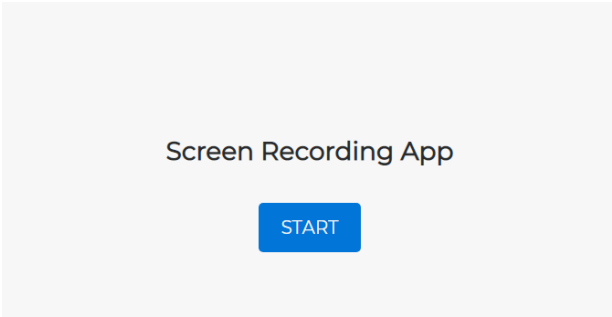

Рисунок 2.3 — Початкова сторінка рекордера екрану

Відеоредактор розроблено як односторінковий додаток. У майбутньому сайт буде готовий до розширення за допомогою додаткових опцій, таких як запис веб-камери окремо так і разом з екранним простором.

Обов’язково буде надано вибір частини екрану для запису, де можемо обрати чи записувати весь екран, чи певне вікно або вкладку браузера, а також надаватиметься дозвіл на використання мікрофону. При чому, дозвіл на використання мікрофону здійснюється лише один раз, а на захват екрану — при кожному запуску запису.

Після виконання запису, повинно бути відображено вікно перегляду зібраного матеріалу, і користувач матиме змогу завантажити його файл на свій пристрій, приклад показано на рис 2.4.

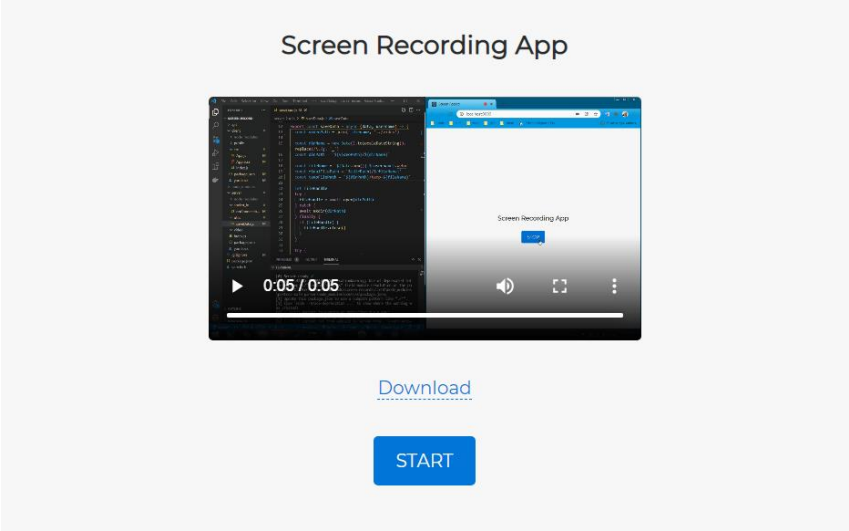

Рисунок 2.4 — Екран користувача рекордера екрану

У вікні буде надано плеєр для перегляду записаного матеріалу. Керування переглядом здійснюється за допомогою кнопок, які прикріплені до нижньої частини області відео (відтворення/пауза, гучність, перехід до повноекранного режиму, опції і таймлайн відео). При цьому програвач буде відтворювати лише ті кадри які були записані останніми. Для проведення швидкого перегляду матеріалу можна натискати на часову шкалу, тим самим виконуючи стрибок до конкретного кадра відео. Приклад роботи рекордера екрану описано в таблиці 2.1.

Таблиця 2.1 — Текстова специфікація випадку використання екранного рекодера

Випадок використання: записати екран, завантажити відео на комп’ютер
ID: User1
Учасники: Користувач, Система
Умови: Користувач відкрив веб-сайт із рекордером.
Хід подій: 1. Користувач натискає на кнопку початку запису 2. Вибирає опції запису 3. Виконує запис матеріалу 4. По закінченню запису, користувач переглядає матеріал і завантажує на пристрій

Після завантаження початкової сторінки відеоредактора з’являється можливість створити новий проект. У майбутньому сайт буде готовий до розширення за допомогою додаткових опцій, таких як шаблони, аутентифікація користувачів або список користувацьких проектів. Щоб почати редагування, користувач вибирає «Create new project», рисунок 2.5.

Після створення нового проекту користувач перебуватиме в інтерфейсі створення відео. Відеоредактор розроблено як односторінковий додаток. Користувач постійно працює на одному екрані, і навіть під час діалогових вікон редактор закривається лише частково, так що користувач відчуває, що він весь час контролює свій проект.

Рисунок 2.5 — Екран привітання для створення нових проектів

При розробці інтерфейсу користувача редактора мене найбільше надихнули настільний додаток iMovie, Shotcut та онлайн-інструмент для редагування фотографій — Photopea.

Інтерфейс користувача редактора розділений на 4 частини. Макет був розроблений та протестований на ексізі, що показує рис. 2.6.

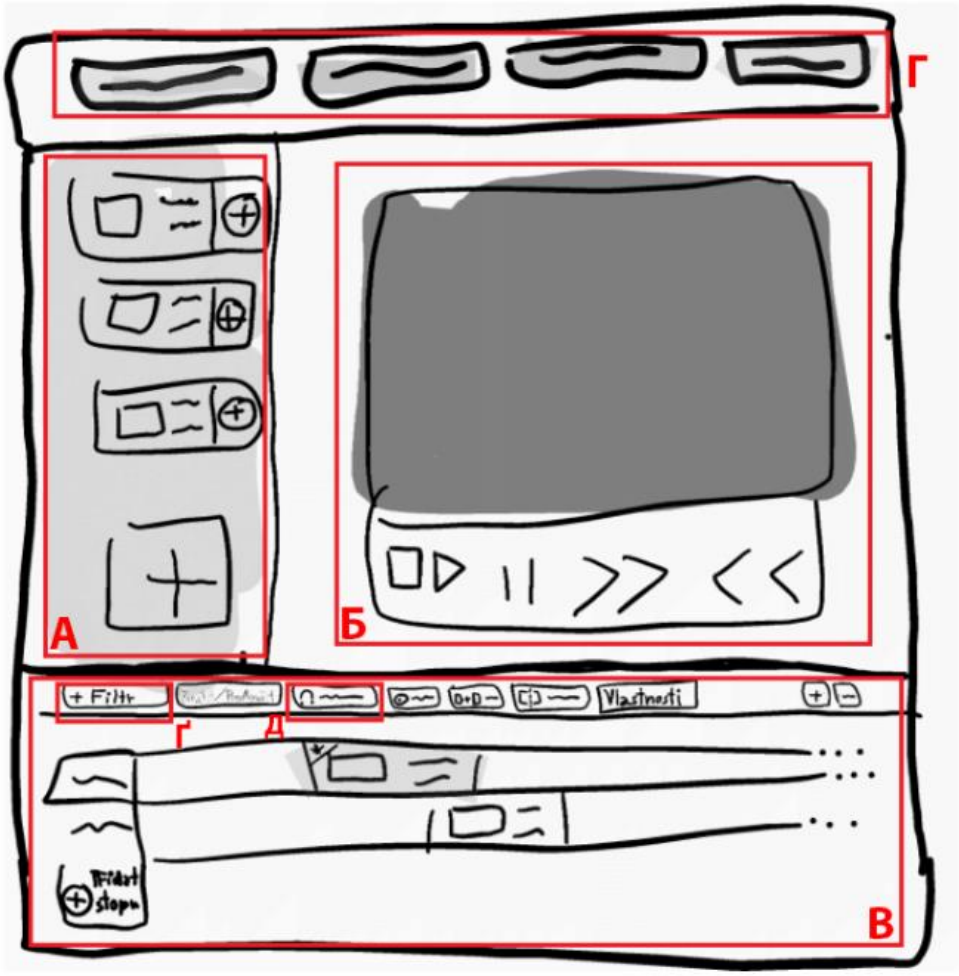

Рисунок 2.6 — Ескіз інтерфейсу користувача з позначеним розташуванням елементів керування

На основі моделі виготовлено прототип основного інтерфейсу користувача відеоредактора, який відображено на рис. 2.7.

У верхній частині екрана розташована панель інструментів дій над проектом (див. рис. 2.6) — скасувати редагування або відтворити проект. У майбутньому з’явиться можливість перемикання мови і можливість експортувати або видаляти дані.

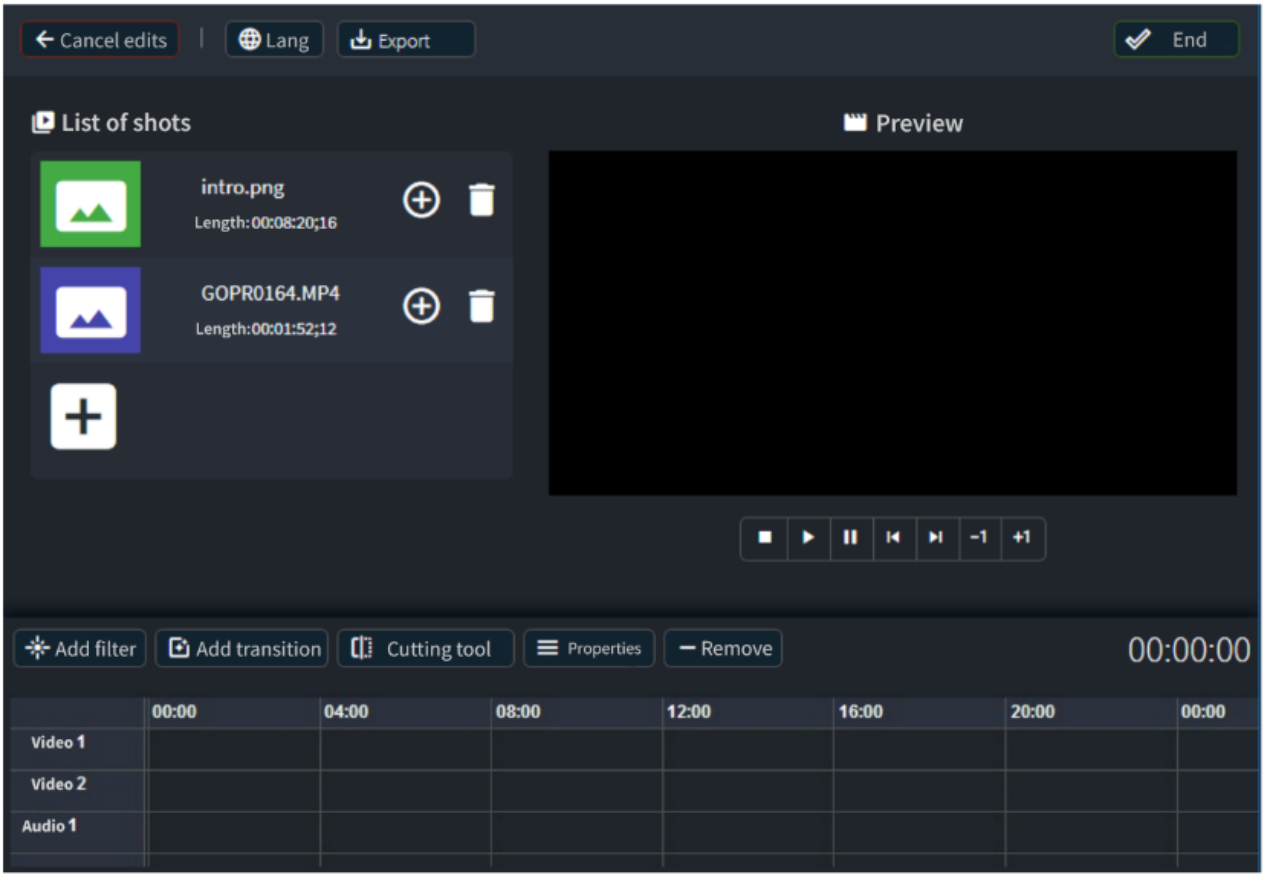

Рисунок 2.7 — Прототип інтерфейсу користувача. Він заснований на сітчастій моделі.

Операції над відео поділено на 3 частини, що відповідають макету інтерфейсу користувача. У лівій частині макету (див. рис. 2.6) знаходиться список вихідних матеріалів (відео, малюнки тощо), які можна використовувати в проекті.

Елементи списку вихідних матеріалів обробляються безпосередньо. Доступні операції відображаються постійно.

Під останнім пунктом є кнопка для додавання матеріалу, або можна скористатися перетягуванням файлу в цю область. Зображення, відео та аудіофайли можна завантажувати в список джерел. Підтримувані формати відео/аудіофайлів такі ж, як і FFmpeg. З файлів зображень підтримуються формати зображень PNG і JPEG.

На рис. 2.8 показана схема варіантів використання для роботи з вихідним матеріалом.

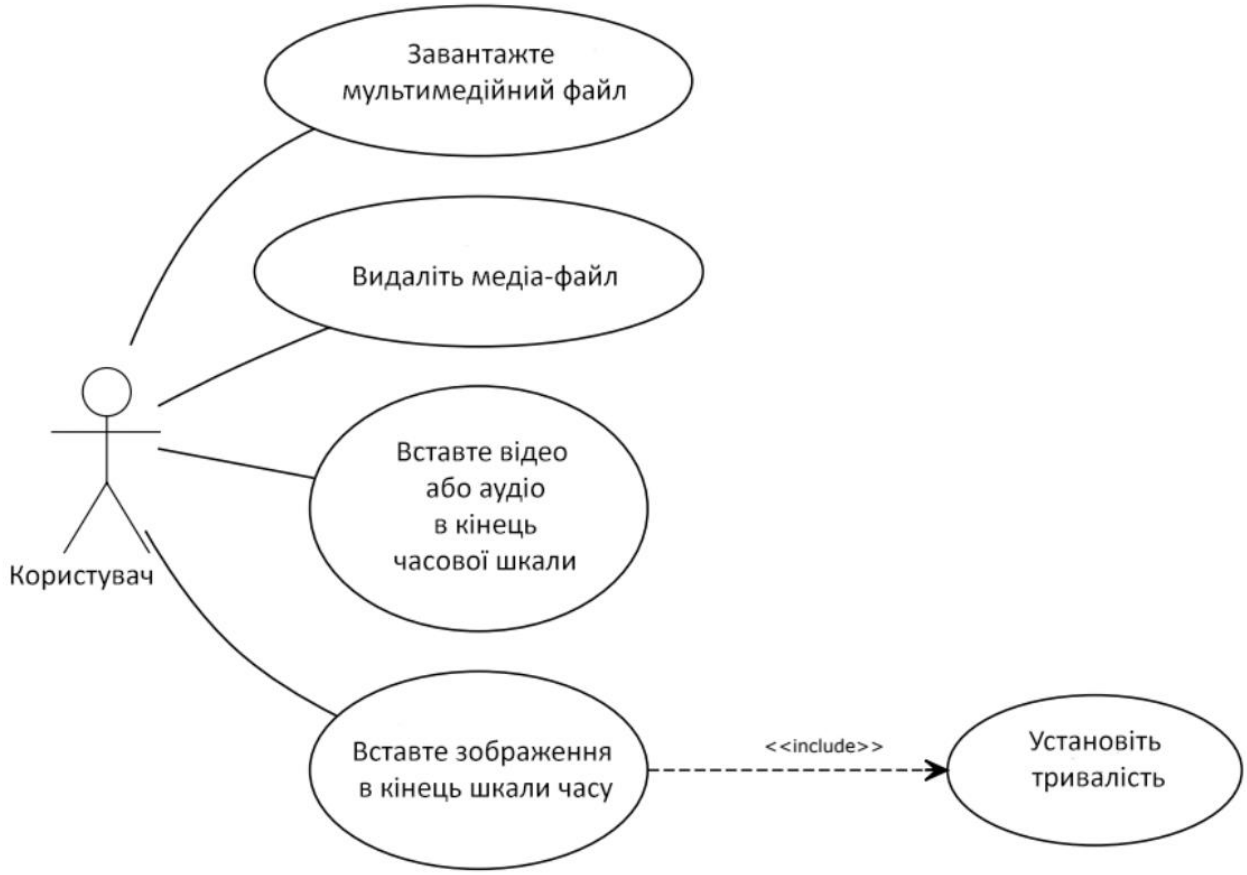

Рисунок 2.8 — Схема варіантів використання для роботи з ресурсами у відеоредакторі

У правій половині макету (див. рис. 2.6) знаходиться програвач попереднього перегляду отриманого відео. Управління програвачем попереднього перегляду здійснюється за допомогою кнопок, розташованих безпосередньо під програвачем (відтворення, зупинка, перехід до початку або кінця, перехід до початку або кінця поточного елемента). Програвач відтворює лише файли, розташовані на часовій шкалі, він не відображає попередній перегляд вихідного матеріалу. Поточна позиція програвача вказується вказівником на часовій шкалі. Відтворення можна розпочати на початковій швидкості та призупинити. Щоб маніпулювати позицією у відео, використовується стрибок вперед/назад, який переміщує вказівник на найближчий початок або кінець елемента будь-якої доріжки, діаграма варіантів використання показана на рис. 2.9.

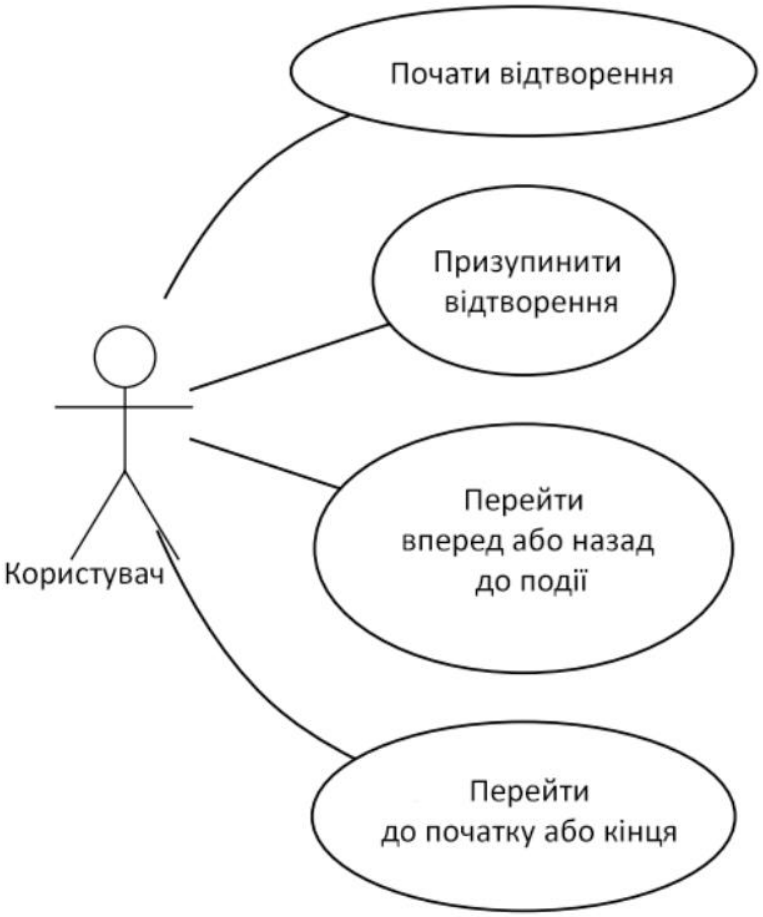

Рисунок 2.9 — Схема варіантів використання для роботи з програвачем попереднього перегляду

В нижній частині макету (див. рис. 2.6) є часова шкала з аудіо- та відеодорожками. На часовій шкалі відображаються композиції в порядку їх створення. Доріжки додаються та видаляються за потреби. Якщо користувач захоплює елемент і перетягує його нижче останньої відео/аудіодоріжки, створюється новий. Оригінальний дизайн передбачав відображення назв треків і додавання треків вручну. Після тестування я вирішив запропонувати робочий стіл, на якому він може переміщувати елементи на доріжках. Елементи додаються до шкали часу зі списку матеріалів за допомогою кнопки «+» поруч із матеріалом. Переміщення елементів на часовій шкалі здійснюється шляхом захоплення елемента та перетягування в нове місце, переміщення до іншої часової шкали здійснюється перетягуванням на іншу вісь того ж типу. Ви можете змінити початок або кінець, захопивши елемент за його край. Кожен елемент має

зображення попереднього перегляду та текстову інформацію про елемент. Ви можете працювати з вибраним елементом за допомогою кнопок на панелі інструментів у верхній частині часової шкали. У програмі є дві панелі інструментів: глобальні інструменти, що знаходяться у верхній частині макету (див. рис. 2.6), та інструменти шкали часу, знаходяться над шкалою часу згідно макету (див. рис. 2.6). Якщо до елемента застосовано хоча б один фільтр, у верхньому лівому куті з’явиться «наклейка» зі значком фільтра. Часову шкалу можна збільшити та зменшити за допомогою кнопок на панелі інструментів шкали часу.

Якщо шкала часу збільшена так, що не можна відобразити всі елементи шкали часу, ви можете перемістити вісь у сторони, захопивши порожній простір на доріжці та перетягнувши у сторони. У новому проекті доступна порожня аудіо- та відеодоріжка. Індикатор поточного часу можна захопити та перемістити. Попередній перегляд відео буде відтворено під час індикатора, і в цей час відео можна редагувати, діаграма варіантів використання — рис. 2.10.

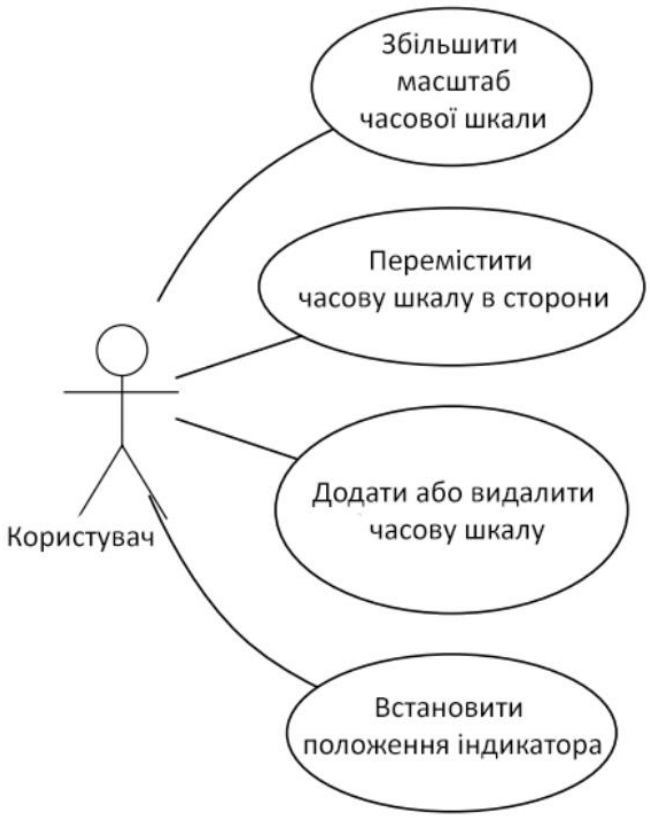

Рисунок 2.10 — Схема варіантів використання роботи з шкалою часу

Після вибору елемента на шкалі часу на панелі інструментів шкали часу доступні наступні дії. Елемент можна видалити, навколишні елементи залишаються, переходи скасовуються. Ви можете додати фільтр зображення (див. рис. 2.6 Е) (текст, накладання HTML, яскравість, контраст, відтінок, яскравість, насиченість, баланс білого, поворот, розмір, положення, обрізання, прозорість, різкість) або аудіофільтр (вимкнути звук, збільшити/зменшити). При додаванні змішаного переходу між 2 елементами (див. рис. 2.6 Д), елементи з переходами ведуть себе як ціле під час переміщення. Ви також можете переглянути підсумкові властивості. Ви можете змінити початок або кінець відео/аудіофайлу або довжину відображення зображення, захопивши край елемента та перетягнувши його в потрібне положення. Дія з вибраним елементом показує діаграму варіантів використання на рисунку 2.11.

Рисунок 2.11 — Діаграма варіантів використання роботи з елементами шкали часу

Якщо вам потрібно вказати, яку дію хоче виконати користувач (вибір фільтра), з’явиться випадаюче меню. Якщо вам потрібно ввести значення (наприклад, фільтр), з’явиться модальне вікно. Модальні вікна також використовуються для відображення властивостей. Довідка для елементів вирішується за допомогою інформаційних бульбашок (іноді також вспливаючих підказок).

Якщо нова система замінить існуючий процес обробки відео, вона повинна мати можливість виконувати всі етапи редагування відео. У таблиці 2.2 подано найпоширеніший варіант використання, коли необхідно обрізати початок і кінець відео та вставити вступний екран із умовами використання перед відео.

Таблиця 2.2 — Текстова специфікація варіанту використання для використання відеоредактора

Випадок використання: вставити вступ, обрізати відео
ID: User1
Учасники: Користувач, Система
Умови: 1. У користувача є картинка із вступом та відео. 2. Користувач відкрив веб-сайт разом із редактором.
Хід подій: 1. Використання починається із завантаження відео та внутрішнього зображення. 2. Користувач вставляє відео та зображення на шкалу часу, встановлює в діалоговому вікні довжину внутрішнього відображення. 4. Користувач відтворює попередній перегляд відео в вказаний час початку та завершення відео, призупиняє відтворення та ділить на 2 частини. 5. Користувач підтверджує зміни та заповнює параметри отриманого відео. 6. Система генерує XML і пропонує його для завантаження або обробки на сервері.

2.3 Дизайн API відеоредактора

Так як інтерфейс відеоредактора є дещо складнішим як і в архітектурі так і в розробці, то потрібно розробити для нього свій API. Як і інтерфейс користувача, інтерфейс сервера потрібно спроектувати та протестувати. Під час проектування необхідно дотримуватися правил архітектури REST. REST визначає яким чином дані можуть оброблятися на сервері. Використовуються 4 способи обробки:

- для створення даних (Create);
- для отримання необхідних даних (Retrieve);
- для зміни даних (Update);
- для видалення даних (Delete).

Цей квартет називається CRUD.

По-перше, необхідно було написати операції, які дозволить API. Такими операціями для відеоредактора є: створити новий проект, отримати статус існуючого проекту, завантажити файл із пристрою, вставити завантажений файл на часову шкалу, додати доріжку, видалити трек, додати фільтр, видалити фільтр, додати перехід, видалити перехід, перемістити елемент, розділити його на 2 частини, видалити елемент із шкали часу, видалити завантажений файл, відобразити проект або видалити проект.

Наступним кроком було визначення ресурсів — сутностей. Однією з сутностей, безсумнівно, буде проект, потім файл і елемент шкали часу. Я вказав перехід, фільтр і відстеження з іншими сутностями. Операції можна виконувати тільки над ресурсами; після визначення ресурсів необхідно призначити операції ресурсам. Я призначив проекту такі сутності: створити новий проект, отримати статус існуючого проекту, відобразити проект, видалити проект. Над файлом можна буде такі дії: завантажити файл з пристрою, вставити завантажений файл на шкалу часу, видалити завантажений файл. Елемент шкали часу дозволить вам: перемістити елемент, розділити його на 2 частини, видалити елемент із шкали

часу. Фільтр можна додавати та видаляти, а також додавати та видаляти перехід і трек.

REST розрізняє 4 операції. Тепер вам потрібно призначити операціям один із наступних методів HTTP:

— POST (Створити);

— GET (Отримати);

— PUT (Оновити);

— DELETE (Видалити).

Нові ресурси створюються при створенні нового проекту, завантаженні файлу з пристрою, додаванні фільтра, переході та треку, для цих операцій був обраний метод POST. Зрозуміло, що отримання статусу проекту є методом GET. Метод DELETE використовується для видалення ресурсів — видалення фільтра, переходу, треку, елемента з шкали часу, видалення завантаженого файлу, видалення проекту. Для решти операцій можна використовувати метод PUT — вставити записаний файл на шкалу часу, перемістити елемент, розділити його на 2 частини.

В ідеалі колізії не відбуватимуться, і кожен ресурс матиме максимум одну операцію, призначену одному методу HTTP. У цьому випадку вихідний елемент має два методи PUT на часовій шкалі — перемістити елемент, розділити його на 2 частини. На цьому етапі вам потрібно ще більше диференціювати операцію, ідентифікувати переміщення і розділити елементи як розділення.

Щоб уникнути конфліктів між URL-адресами веб-сайту та API, усі запити API мають префікс API. Наприклад, це дозволить вам оновити версію API в майбутньому. Отриманий дизайн API наведено нижче:

а) project;

— POST: /project;

— GET: /project/{projectID};

— DELETE: /project/{projectID};

— PUT: /project/{projectID};

б) file;

— POST: /project/{projectID}/file;

— DELETE: /project/{projectID}/file/{fileID};

— PUT: /project/{projectID}/file/{fileID};

в) filter;

— POST: /project/{projectID}/filter;

— DELETE: /project/{projectID}/filter;

г) transition;

— POST: /project/{projectID}/transition;

— DELETE: /project/{projectID}/transition;

д) track;

— POST: /project/{projectID}/track;

— DELETE: /project/{projectID}/track;

е) item;

— DELETE: /project/{projectID}/item;

— PUT: /project/{projectID}/item/move;

— PUT: /project/{projectID}/item/split.

Запит REST API має бути повним, REST API не має стану. Наприклад, коли запит DELETE /project/{projectID} завершено, ми кажемо серверу видалити проект, а також вказуємо ідентифікатор проекту. Однак для деяких вимог необхідно вказати вимоги. Наприклад, для запиту POST: / project / {projectID} / filter ми не знаємо, до якого фільтра застосувати, і не знаємо, до якого елемента. За винятком методу GET, вимоги можуть мати тіло з параметрами. Параметри можна передавати в будь-якому форматі, я використовую в проекті application/json і для завантаження багатокомпонентного/форма-файлу. Попередня вимога визначається параметрами track, item, filter і params (параметри фільтра). Повну специфікацію запропонованого API можна знайти в документації API. Я задокументував API відповідно до специфікації OpenAPI 3.0 за допомогою онлайн-інструменту Swagger Editor.

3 ФУНКЦІОНАЛЬНЕ ПРОЕКТУВАННЯ ВЕБ-ДОДАТКУ

3.1 Структура проекту

Кореневий каталог містить файли проекту веб-додатку, що складається з: головної сторінки, також папки субдодатків — рекордера екрану та відеоредактора. Загальна структура показана на рис. 3.1.

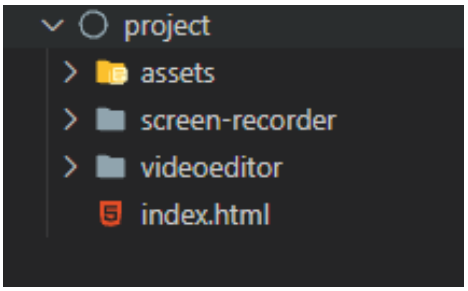

Рисунок 3.1 — Загальна структура проекту

В каталозі програми для запису екрана знаходяться файли конфігурації в форматі JSON, а також підкаталоги із клієнтською та серверною частиною, згілно клієнт-серверної архітектури. Структура каталогу рекордера екрану показана на рис. 3.2.

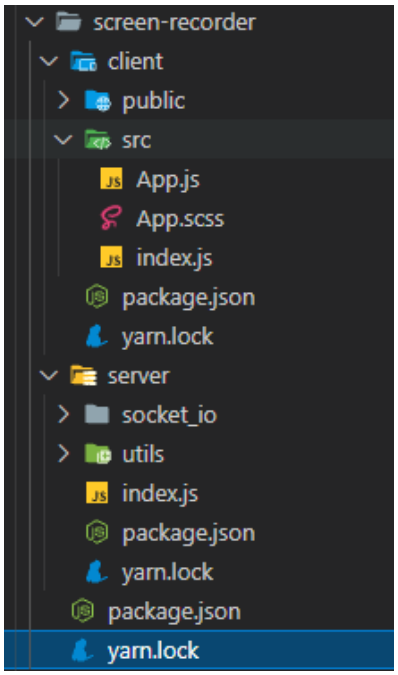

Рисунок 3.2 — Структура субдодатку рекордера екрану

У папці відеоредактора містяться файли конфігурації, файли менеджера пакетів і основний файл запуску проекту. Вихідні файли зберігаються в підкаталогах, архітектура MVC очевидна зі структури проекту. Структура субдодактку відеоредактора наведена у рис. 3.3.

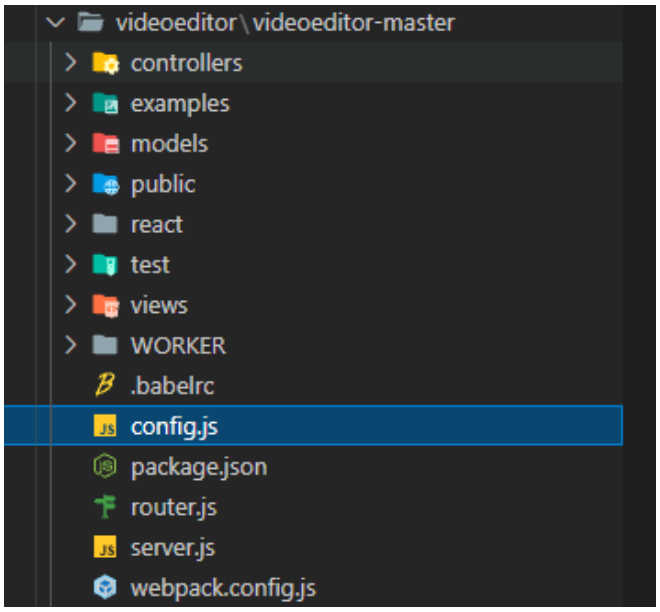

Рисунок 3.3 — Структура субдодатку відеоредактора

Відповідно, також було створено узагальнену функціональну структурну схему веб-додатка. У рисунку Б.1 показано дану структурну схему. Однак, розглянемо детальніше.

Дана структурна схема включає в себе клієнта — користувача, елемента Dropzone, а також і такі файли:index.html головної сторінки та сторінки рекордера екрану, style.css, index.js для клієнта і сервера рекордера екрану, App.js, App.scss, onConnection.js, saveData.js, front.js, main.html, project.html, Editor.js, style.scss. Timeline.js, Preview.js, SubmitDialog.js, Sources.js, Uploader.js, SourceTableRow.js.

Клієнт — безпосередньо наш користувач, вся відображувана інформа інформація буде надходити до нього з файлів index.html, main.html, project.html. Дані файли містять HTML-код, що призначений для розмітки веб-сторінки.

Файли style.css, App.scss, style.scss, містять код каскадної таблиці стилів (CSS), причому оскільки останні 2 файла використовуються у відеоредакторі та

рекордері екрана, то було вирішено використати SCSS (Sassy Cascading Style Sheets) — більш просунутий варіант CSS, використовує мову Ruby,. Його також називають Sassy CSS через його розширені функції. Це мова попереднього процесора, яка компілюється або переривається в CSS. Ми можемо додати кілька додаткових функцій до CSS за допомогою SCSS, включаючи змінні, вкладення та багато іншого. Усі ці додаткові функції можуть зробити написання SCSS набагато простішим і швидшим, ніж написання стандартного CSS. SCSS може використовувати код і функцію CSS. SCSS повністю відповідає синтаксису CSS, хоча також підтримує повну потужність SASS.

У App.js повинно відбуватись наступне: розмітка сторінки, запис кадрів, відправка даних на сервер, отримання з сервера даних. Саме тому для цього у ньому надається URL сторінки клієнта — http://localhost:4000, що працює на протоколі HTTP, а також є функція, що відслідковує підключення і отримання джерела запису (відео, аудіо), функції початку та завершення запису, які прив’язані до обробника подій і також значення, що має повертати — HTML-код і дані для інших функцій/файлів.

У index.js сервера рекордера екрана відбувається створення сервера на основі Socket.io із присвоєнням йому URL, причому там також вмикається з’єднання та у файлі onConnection.js перевіряється підключення користувача. Зокрема у файлі saveData.js відбувається обробка записаних кадрів, після чого через ffmpeg відбувається об’єднання в 1 файл із певною конфігурацією (див. стор. 78).

У файлі front.js відбувається front.js прив'язка React до HTML, відображення початкової сторінки — при перевірці повинно відобразити сторінку — main.html, до якої прив’язана розмітка із файлу newProjectDialog.js — діалогове вікно із створенням нового проекту, та project.html, до якого прив’язана розмітка із файлу Editor.js.

Опис компонентів файлів — Editor.js, Timeline.js, Preview.js, SubmitDialog.js, Sources.js, Uploader.js, SourceTableRow.js див. стор. 67.

3.2 Розробка відеоредактора

3.2.1 Налаштування, файли конфігурації

Коли програма запускається, починає виконуватися код у файлі /server.js. Спочатку налаштовується середовище обробки запитів. Для цього використовуються додаткові файли - /config.js, /router.js і файли встановлених пакетів.

Файл /server.js є основним файлом проекту. Коли проект запускається, інтерпретатор Node.js починає інтерпретувати цей файл. Файл відповідає за ініціалізацію середовища, зокрема за наступні дії:

— ініціалізує JavaScript-фреймворк Express.

— ініціалізує розширення для обробки завантажених файлів (busboy).

— ініціалізує розширення для обробки тіл запитів у форматі JSON

— встановлює систему шаблонів від експрес-подання до чистого HTML у папці /views.

— встановлює маршрутизатор на /router.js

— встановлює /public як загальнодоступний каталог, доступний з мережі.

— налаштовує сервер на прослуховування порту та адреси, зазначених у файлі config.js.

Ці дії виконуються лише при запуску сервера. Після виконання всіх дій сервер прослуховує запити та пересилає вхідні запити на маршрутизатор.

Файл /config.js є центром усіх директив конфігурації. Перед розгортанням у новому середовищі необхідно налаштувати значення у файлі /config.js відповідно до поточних даних. Саме сім рядків даних залежать від середовища, в якому планується розгортання:

— port —номер порту, на якому буде працювати сервер, причому порт повинен бути вільним та для портів з номером менше 1024 можуть знадобитися права адміністратора;

— host — доменне ім'я, наприклад, "localhost", яке є значенням за замовчуванням;

— emailServer — доменне ім'я SMTP-сервера для відправки електронних повідомлень;

— emailPort — номер порту SMTP-сервера для відправки електронних повідомлень, порт за замовчуванням «465».

— emailUser — ім'я користувача облікового запису для аутентифікації на SMTP-сервері, а сповіщення електронною поштою надсилаються з цього облікового запису;

— emailPasswd — пароль облікового запису для аутентифікації на SMTP-сервері.

— adminEmail — електронна адреса адміністратора, дана адреса використовується для надсилання помилок адміністратору, для зв'язку.

Нижче наведено запис projectPath, який визначає розташування підкаталогу MLT проектів. Підкаталоги містять як завантажені файли, так і файл XML для програми MLT. Вам не потрібно змінювати значення за замовчуванням "WORKER", якщо, наприклад, вам не потрібно мати файли на іншому диску.

Останнім елементом є об’єкт mapFilterNames. Об’єкт використовується для зіставлення імен користувацького фільтра (які не є частиною MLT) для фільтрування імен, які є частиною MLT. Вам не потрібно змінювати об’єкт, якщо ви не реалізуєте додатковий псевдонім для імен фільтрів.

Директиви конфігурації також містять функції serverUrl і apiUrl, які спрощують доступ до адреси в проекті. Значення використовуються, наприклад, при створенні асинхронних запитів у компонентах React. Функції повертають значення для протоколу HTTP, у разі розгортання HTTPS необхідно змінити ці дві функції.

Маршрутизація забезпечується файлом router.js, який складається з правил маршрутизації та проміжного програмного забезпечення. Правила перекладають URL-адресу у функцію JavaScript (контролер), яка обробляє запит. З точки зору архітектури MVC, роутер також належить до контролерів. Правило

маршрутизації зазвичай застосовується першим, відповідно до URL-адреси запиту, правила проміжного програмного забезпечення виконуються перед правилами маршрутизації, і якщо ми викликаємо наступну функцію в кінці, наступні правила також виконуються. Правило проміжного програмного забезпечення використовується для реєстрації всіх доступів, у майбутньому воно також використовуватиметься для авторизації користувачів. Обробник запиту отримує об’єкт запиту, майбутній об’єкт відповіді та наступну функцію. У лістингу 3.1 наведено першу функцію — це проміжне програмне забезпечення, а другу — маршрутизація.

Лістинг 3.1 — Реалізація функції запису доступу та маршрутизації

```
router.use((req, res, next)=>{
        console.info(new Date(),‘ @ ${req.originalUrl}‘);
        next();
});
router.post(’/api/project/:projectID/file’, apiController.projectFilePOST);
```

Якщо маршрутизатор не знаходить відповідне правило маршрутизації в списку правил, він повертає відповідь з кодом 404. Тіло відповіді містить HTML з текстом «Cannot GET / foo» для запиту методу «GET» з адресою "/ foo".

Останнє, що потрібно — це перехоплення помилок, яке використовується, якщо під час належної обробки запиту виникає помилка і викликається наступна функція. Помилка обробляється errorController викликом такого методу — router.use(errorController.default);

Файли /package.json і /package-lock.json використовуються для зберігання інформації про проект. Файли використовуються програмою npm. У файлі /package.json зберігається така інформація про проект, як назва та опис проекту, версія, репозиторій GitHub, інформація про автора та ліцензію, URL-адреса звіту про помилку та URL-адреса сторінки проекту. Тут також зберігаються назва основного файлу JavaScript (server.js) і скрипти. Наприклад, якщо ви введете npm run dev-build у кореневому каталозі, буде виконана команда ./node_modules/.bin/webpack -wd.

Найважливішою частиною файлу є список залежностей. Якщо нам потрібно розширити функціональність проекту та інші функції, ми можемо скористатися системою пакетів npm1. Станом на 22 квітня 2019 року в системі було зареєстровано майже 810 тис. пакетів. Менеджер npm — це стандартна система пакетів Node.js, подібна до менеджера залежностей Composer для PHP. Використовуйте команду npm install —save, щоб завантажити та встановити залежність, і в той же час ця залежність зберігається у файлі /package.json.

Для простоти я не вказую файл /package-lock.json у структурі проекту. Цей файл не потрібен для роботи програми. Якщо файл не існує, npm створить його. У цьому файлі зберігається поточний стан інстальованих пакетів, а в /package.json — список пакетів, які «повинні» бути встановлені. Наприклад, якщо потрібно встановити пакет React версії 16.8.6, то фактично можна встановити пакет версії 16.8.9. Оскільки залежності діють лише від основної версії, можна встановити нові версії, сумісні зі зворотним зв’язком (наприклад, виправлення помилок).

3.2.2 Патерн MVC: Модель

Модель — це частина сервера, яка відповідає за роботу з даними, операціями з файлами, а також надає, наприклад, послуги електронної пошти. Ця частина закрита від запитів і відповідей, окремі модулі можна використовувати окремо, необхідні дані передаються функціям у параметрах. Функції можуть повертати або не повертати значення.

Модуль emailManager містить функцію sendProjectFinished. Ця функція використовується після завершення рендерингу отриманого відео. Як параметри потрібна адреса електронної пошти одержувача або кілька адрес, розділених комами, ідентифікатор проекту та результат обробки (істина для успіху, false для помилки). Функція надсилає електронні листи через SMTP-сервер у конфігурації. В результаті додатку не потрібен власний сервер електронної пошти. У разі успіху електронний лист буде надіслано лише власнику проекту. Якщо під час обробки виникає помилка, електронний лист надсилається як

власнику, так і адміністратору. Бібліотека nodemailer використовується для надсилання електронних листів.

Модуль fileManager відповідає за отримання інформації про завантажені файли. Він містить функцію getDuration, яка є асинхронною. Функція повертає Promise. Якщо ми запитаємо довжину відео- чи аудіофайлу, буде створено нащадок процесу, і в ньому буде запущена команда FFmpeg, щоб визначити довжину носія. Після того, як довжина знайдена, повертається значення функції розв'язування. Запит для іншого типу файлу повертає null.

Модуль mltxmlManager полегшує роботу з файлом XML, створеним для програми MLT. Модуль відповідає за зберігання файлу project.mlt. В якості параметрів він вимагає ідентифікатор проекту і вміст кореневого елемента, включаючи сам тег. Функція заповнює заголовок XML і записує весь рядок у файл проекту. Він також містить функції для отримання відносного шляху до файлу MLT (getMLTpath) і до каталогу проекту (getWorkerDir). Решта функції працюють безпосередньо з XML, спрощуючи створення, наприклад, списків відтворення.

Модуль timeManager використовується для роботи з мітками часу. У проекті я використовую час у форматі «00:00:00,000» для позначення моментів часу та для вказівки тривалості, причому тривалість не повинна бути нульовою. Цей модуль містить функції для додавання та віднімання часу, для зменшення часових інтервалів вдвічі та перевірки правильності формату тривалості (isValidDuration). Модуль використовується як на стороні сервера, так і в компонентах React.

Модулі охоплюють пов'язані функції в один блок. Модулі містять змінні та функції, які стають доступними за допомогою імпорту. Щоб мати можливість імпортувати їх, необхідно вказати ключове слово export в модулі. Якщо в модулі є більше однієї функції, ми повинні вказати, яку функцію ми хочемо імпортувати. Якщо не вказано, імпортується експорт за замовчуванням. Якщо ми хочемо використовувати більше функцій з одним імпортом, необхідно обернути всі функції та змінні в модулі в загальний об'єкт, який буде експортувати за

замовчуванням. У лістингу 3.2 використовується один основний об’єкт, який є експортом за замовчуванням.

Лістинг 3.2 — Реалізація експорту за замовчуванням

```
export default{
        subDuration(durationA, durationB){
                //…
                return subResult;
        },
        addDuration(durationA, durationB) ){
                //…
                return subResult;
        },
        //…}
```

Якщо ми хочемо використовувати модулі, процедура виконується як на стороні сервера, так і на стороні клієнта (у React), що показано у лістингу 3.3.

Лістинг 3.3 — Приклад використання модуля

```
import timeManager from ’../../models/timeManager’;
let actualTime = timeManager.addDuration(timeA, timeB);
```

Модулі використовуються в рамках одного проекту, вони не є пакетами npm. Вони є новішою альтернативою використанню команди require.

Завдяки подіям та асинхронному доступу до блокуючих запитів, не проблема чекати набагато складніших операцій, ніж робота з файловою системою. Проект потребує обробки, перетворення та отримання мультимедійної інформації. Для JavaScript він може створити дочірню частину, яка обробляє відеофайл і працює з його виводом, коли дочірня частина закінчить. Це було б проблемою з PHP, і незавершені процеси можуть призвести до вичерпання ресурсів сервера. Модуль Child Processes (child_process) робить доступною роботу з процесами. З нього я використовую функцію exec, яка створює оболонку і дозволяє виконувати в ній команди. Після виконання останньої команди буде доступний вміст стандартного виводу та стандартного виводу помилок. Лістинг 3.4 демонструє використання команди exec.

Лістинг 3.4 — Приклад використання команди exec

```
exec(‘ffmpeg −i ${filepath} 2>&1 | grep Duration | cut −d \’ \’ −f 4 | sed s/,// |
sed s/\\\\./,/‘,
        (err, stdout, stderr) => {
        if (err) console.error(err);
        else {
                console.log(stdout.trim());
        }
        });
```

Функція exec створює дочірній елемент і отримує в ньому інформацію про довжину файлу. У цьому прикладі він друкує довжину на екрані, але за допомогою Promises ви можете створити функцію getDuration і повертати довжину асинхронно.

Під час візуалізації, файл обробки перевіряється, щоб перевірити, чи вже існує файл обробки в проекті. Щоб відобразити проект подібним чином, використовуючи exec, він викликає програму mlt, приклад реалізації викладено у лістингу 3.5. Після завершення прапорець для обробки видаляється, і надсилається електронний лист про успіх. Якщо виникає помилка, надсилається повідомлення про помилку. Стандартний вихід і стандартний вихід помилок перенаправляються до файлів stdout.log і stderr.log.

Лістинг 3.5 — Реалізація відображення проекту

```
exec(‘cd ${projectPath} && melt project.mlt −consumer
        avformat:output.mp4 acodec=aac
vcodec=libx264 > stdout.log 2> stderr.log‘, (err) => {
        if (err) console.error(‘exec error: ${err}‘);
        fs.unlink(path.join(projectPath, ’processing’), (err) => {
                if (err) console.error(err.stack);});
        if (isset(req.body.email)) {
                emailManager.sendProjectFinished(req.body.email,
                        req.params.projectID, !(err);}
        }
        );
```

Файли XML використовуються як в моделі, так і в контролері. Оскільки це робота з даними, структура створених файлів описана в цьому розділі. Структура зображена графічно на рис 3.3.

Коли створюється новий проект, створюється файл project.mlt, який містить заголовок XML і посилання на схему DTD, розташовану в репозиторії проекту MLT. Після цього використовується кореневий тег. Всередині є один список відтворення для стандартної доріжки videotrack0 і основний контейнер, що містить посилання на всі треки в проекті. Цей основний контейнер завжди має бути останнім елементом у корені. Наступний зразок коду у лістингу 3.6 демонструє вміст файлу project.mlt після створення нового проекту.

Лістинг 3.6 — Демонстрація вмісту project.mlt після створення проекту

```
<?xml version="1.0" encoding="UTF−8"?>
<!DOCTYPE mlt SYSTEM "link”>
<mlt>
        <playlist id="videotrack0"/>
        <tractor id="main">
        <multitrack>
                <track producer="videotrack0"/>
        </multitrack>
        </tractor>
</mlt>
```

Усередині кореневого елемента є перші вихідні файли (елементи), для яких зазначаються такі властивості - абсолютний шлях до файлу (resource), тип файлу (musecut: mime_type), оригінальне ім'я файлу (musecut: name) і у випадку відео довжина файлу в мілісекундах (length). Вихідні файли мають ідентифікатор із префіксом «producer» і 20 символів, які генеруються випадковим чином під час завантаження файлу.

Допоміжні списки відтворення розміщуються після останнього елемента. Допоміжні списки відтворення — це контейнери для елементів фільтра та переходу. У цих списках відтворення завжди є один елемент. У разі списку відтворення цьому елементу може передувати елемент переходу. Допоміжні списки відтворення визначаються як «список відтворення» та серійний номер списку відтворення. Нові списки відтворення вставлені на початку цього розділу.

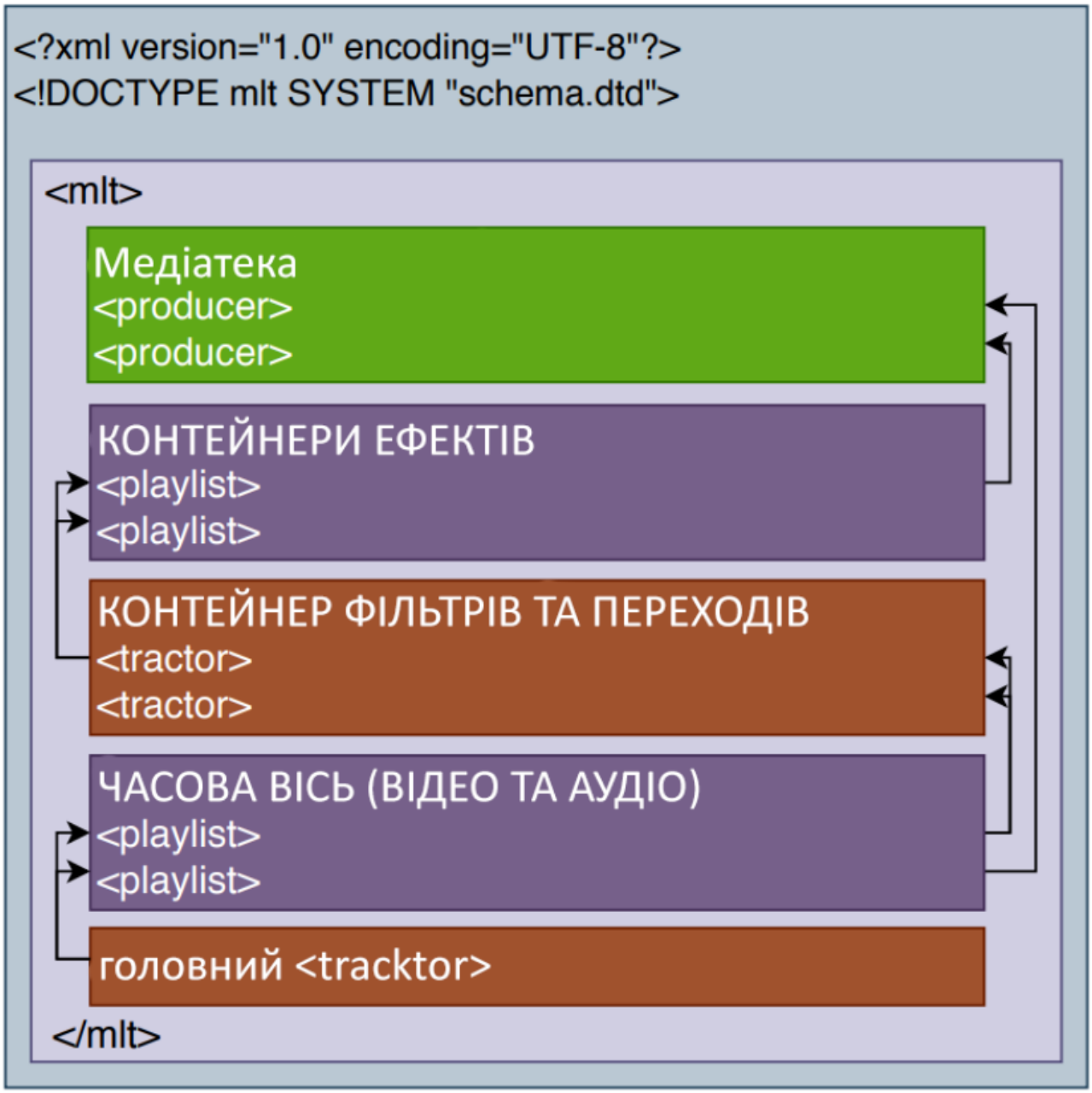

Рисунок 3.3 — Загальна схема елементів створених файлів з позначеними посиланнями

За плейлистами йдуть трактори (елементи <tractor>). Ці елементи являють собою контейнери для фільтрів і переходів. Вони несуть інформацію про застосовані фільтри та переходи. Застосовуйте переходи та фільтри до допоміжних списків відтворення, визначених вище. Трактор може містити як фільтри, так і переходи. Вони ідентифіковані як «tracktor» із серійним номером.

Нові трактори вставлені в кінці поля/перед кінцем з ідентифікатором "videotrack0";

Передостанній розділ у створених файлах є розділом для треків. Треки реалізовані у вигляді списків відтворення. Є відео- та аудіотреки. Відеодоріжки ідентифікуються як «відеодоріжка» і серійний номер відеодоріжки, аудіодоріжки як «аудіодоріжка» і серійний номер аудіодоріжки. Списки відтворення містять послідовність елементів шкали часу. Вони можуть містити посилання на, у випадку простого елемента, або посилання на, у разі застосованого фільтра чи переходу. Інтервал між елементами вирішує вставлення елемента між елементами.

Останній елемент у кореневому елементі завжди повинен бути основним трактором з ідентифікатором "main". Цей трактор містить посилання на всі гілки. Цей елемент лише один і не може бути видалений. Він завантажується програмою MLT, до всіх інших елементів можна потрапити з цього елемента за посиланнями.

3.2.3 Патерн MVC: Вид

Вид — це частина, яка відповідає за відображення HTML-сторінок користувачеві. Він не використовується для запитів API, він призначений для машинної обробки, а не для спілкування з кінцевим користувачем. Знаходиться в папках /views та /react, де У папці /views є шаблони, а папка /react містить компоненти React.

Framework Express надає інструменти для використання різних типів шаблонів. Оскільки для створення інтерфейсу користувача використовуються компоненти React, я вибрав прості файли HTML. У цих файлах є фреймворк для відображення компонентів React у DOM HTML. Файли також посилаються на стилі CSS і файл JavaScript, який містить елементи React. На даний момент використовуються 3 шаблони — main.html, project.html, finished.html. Шаблон main.html використовується для відображення заставки. Містить підготовку до модального вікна для створення нового проекту. Шаблон project.html забезпечує

основу для сторінки відеоредактора. Шаблон finished.html використовується для відображення екрана під час обробки відео. Сторінка періодично оновлюється. Якщо користувач переглядає сторінки без увімкненого JavaScript, буде показано процедуру вирішення проблеми завантаження, вигляд якого представлено на рис. 3.4.

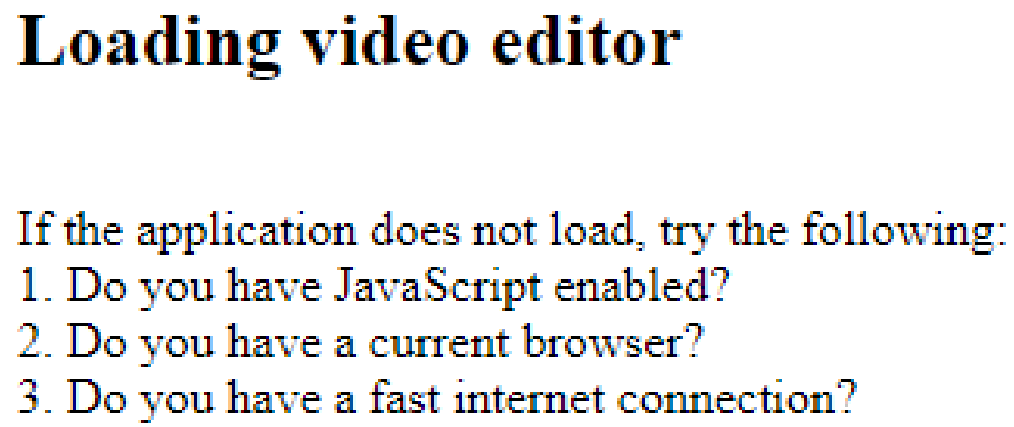

Рисунок 3.4 — Приклад відображення шаблону із вимкненим JavaScript.

Компоненти React використовуються на домашній сторінці для створення проекту та на сторінці з самим редактором. Компоненти для домашньої сторінки знаходяться в папці /react/newProject, компоненти редактора в папці /react/edit.

На домашній сторінці кореневим компонентом є клас NewProjectDialog. Клас відображає діалогове вікно для створення нового проекту. Після натискання кнопки «Створити новий проект» надсилається запит POST /api/project. Якщо API повертає ідентифікатор створеного проекту, він перенаправляється на сторінку з редактором. Клас NewProjectDialog залежить від модального компонента пакета npm react-modal6. Огляд компонентів та їх взаємозалежностей наведено на рис. 3.5.

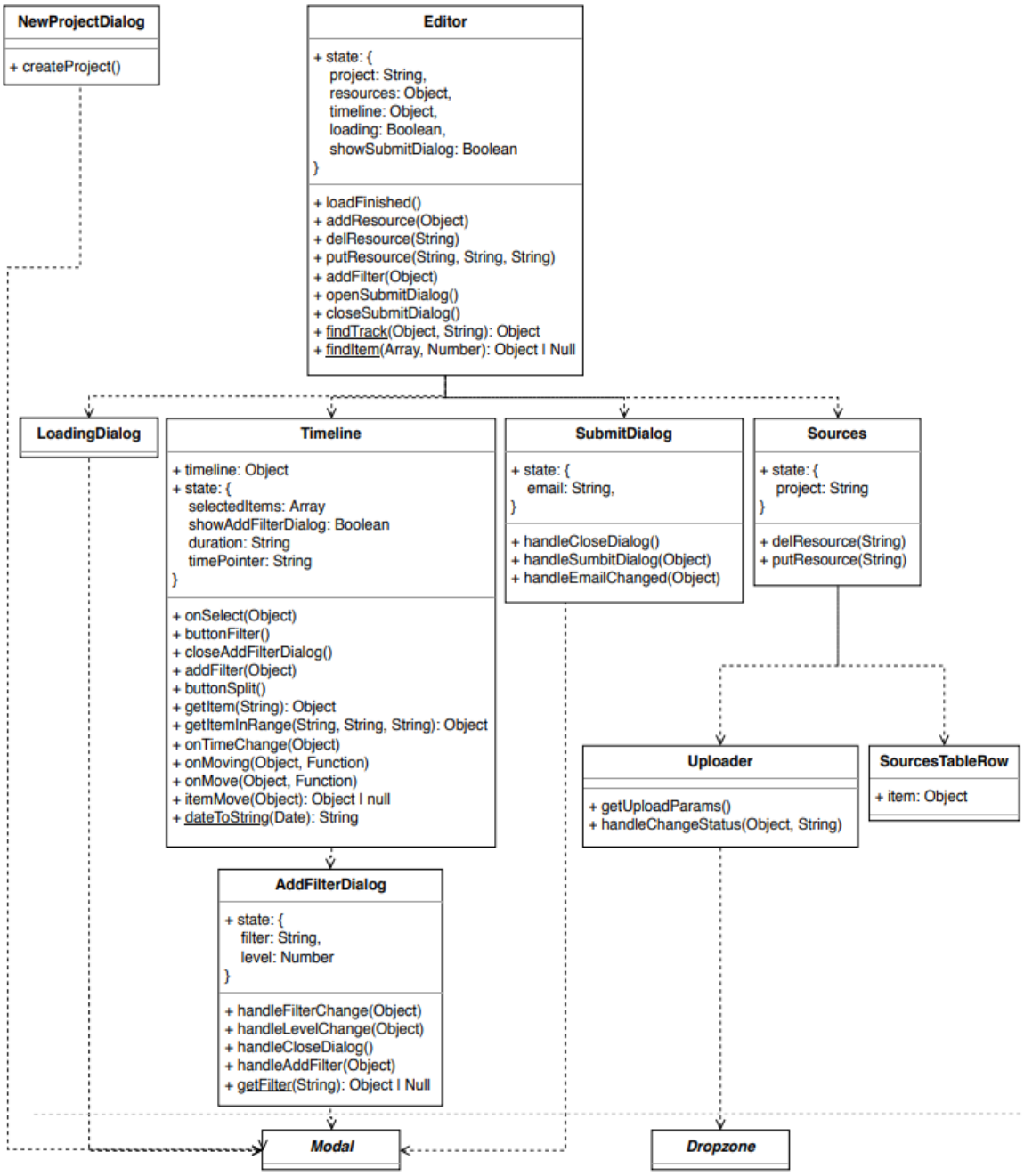

Рисунок 3.5 — UML-діаграма класів, де нащадки класу Component із залежностями типу «use».

Структура компонентів редактора складається з наведених нижче елементів.

Editor — є кореневим компонентом. У своєму складі він містить інформацію про ідентифікатор проекту, список ресурсів проекту та часову шкалу проекту. Отримує статус проекту після завантаження сторінки з API із запитом GET /api/project/:projectID. Після того як він успішно отримує статус проекту, він

приховує оверлей із завантаженням. Він також контролює видимість діалогового вікна візуалізації проекту. Статус проекту поширюється на підкомпоненти Timeline та Sources. Повернені дочірні елементи використовують зворотні виклики для зв'язку з кореневими компонентами, коли проводяться запити на маніпулювання ресурсами та часовою шкалою.

SubmitDialog відповідає за діалогове вікно візуалізації проекту. Містить форму для заповнення електронного листа. Після заповнення адреси електронної пошти та підтвердження він надсилає запит до API PUT /api/project/:projectID. Якщо повертається позитивна відповідь, користувач перенаправляється на сторінку /project/:projectID /finished;

Sources — надає список ресурсів проекту та форму для завантаження додаткових ресурсів. Компонент Sources завантажує файли. Клас Sources використовує дочірній компонент SourcesTableRow для записів вихідного файлу. Якщо дочірні компоненти запитують зміну вихідних файлів/часової шкали, він передає запит до редактора батьківського компонента;

SourcesTableRow малює один рядок у списку медіафайлів. Рядок зберігає інформацію про позицію як свій статус. Ви можете додати елемент ресурсу до шкали часу або видалити його з проекту. Компонент викликає API - щоб додати елемент до шкали часу PUT /api/project/:projectID/file/:fileID і видалити файл DELETE /api /project/:projectID/file/:fileID. Якщо API повертає відповідь без помилок, компонент запитує батьківський компонент Sources змінити вихідні файли/часову шкалу;

Uploader використовується для відтворення форми завантаження файлу. Його завдання — налаштувати компонент Dropzone з пакета npm. Він передає функції компоненту для зворотних викликів, коли статус змінюється (запис завершено, запис збій) і обробляє ці виклики. Коли файл успішно завантажено, він просить батьківський компонент Sources додати файл до списку вихідних файлів;

Timeline, що відповідає за шкалу часу. Часова шкала не є компонентом React. Часова шкала ініціалізується у функції componentDidMount, яка

викликається, коли компонент розміщується у віртуальній DOM. Коли ви змінюєте елементи шкали часу, ви перемалюєте за допомогою функції componentDidUpdate. Компонент також відповідає за переміщення елементів, поділ елементів навпіл і додавання фільтрів, а також відображення діалогового вікна для додавання фільтра — компонента AddFilterDialog. Компонент зберігає поле з позначеними елементами на часовій шкалі та інформацію про те, чи має бути видиме діалогове вікно для додавання фільтра. Якщо діалогове вікно просить вас додати фільтр, запит передається до батьківського редактора;

AddFilterDialog, яктй відтворює форму для додавання фільтра. Він зберігає значення з форми в її стані — вибраний фільтр і його значення. Коли ви перевіряєте форму, компонент створює об’єкт з фільтром та просить батьківський компонент Timeline додати фільтр.

3.2.4 Патерн MVC: Контролер

Перед контролерами стоїть завдання прийняти запит, обробити параметри запиту на основі запиту виконати запитувану дію з використанням моделі, а потім надати відповідь. Контролер може створити відповідь сам або використовувати подання. Контролер керує іншими компонентами. Папка /controllers містить 3 контролери — apiController, errorController і mainController.

Одним із найбільш важливих контролерів є apiController. Він відповідає за весь API. Він відповідає за прийняття запитів, маніпулювання XML і створення відповідей. Цей контролер не використовує представлення, оскільки відповіді API мають формат JSON. У кінці файлу apiController також є 3 допоміжні функції:

— fileErr (створює відповідь з помилкою читання проекту);

— isNaturalNumber (перевірка ці-лих невід’ємних чисел);

— isset (перевірка існування змінних).

Ці функції є допоміжними і використовуються лише у даному файлі.

Найпростіший запит — це GET /api. Сервіс надається функцією за замовчуванням, яка повертає лише інформацію про розташування документації

API. Відповідь генерується в кінці кожної основної функції контролера не пізніше. У разі помилки відповідь генерується відразу після її запису.

Кожна відповідь містить атрибут msg, у разі успіху може містити інші елементи, у разі невдачі відповідь завжди містить err та msg. У наступному прикладі показано, як виглядає відповідь функції, якщо ми не прикріплюємо завантажений файл.

Лістинг 3.7 — Реалізація відповіді функції projectFilePOST

```
res.status(400);
res.json(
{
        err: ‘File missing’,
        msg: “The request body must contain a file to upload.”,
});
return;
```

У разі успіху, запис err буде відсутнім, і, навпаки, відповідь міститиме інформацію про завантажений файл (ідентифікатор файлу, тип і довжина MIME у форматі «00:00:00,00»). Відповідний код наведено у лістингу 3.8.

Лістинг 3.8 — Реалзіація запису err при умові невдачі

```
res.json(
{
        msg: ‘Upload of “${filename}” OK’,
        resource_id: fileID,
        resource_mime: mimeType,
        length: length,
});
```

Для роботи більшості функцій контролера потрібні розширені параметри запиту HTTP. Параметри доступні через об’єкт запиту (перший переданий параметр). Параметри, які є частиною шляху в URL-адресі (projectID, fileID), доступні через об’єкт req.params. Ці параметри завжди визначені. Якщо користувач пропускає параметр в URL-адресі, маршрутизатор не викликає обробник і відображає помилку 404. У деяких випадках неможливо ввести всі параметри безпосередньо в URL-адресу. Параметри також можуть бути в тілі

запиту. Доступ до цих параметрів здійснюється за допомогою req.body. У запиті можуть бути відсутні параметри, контролер повинен спочатку перевірити специфікацію обов'язкових параметрів функцій isset, код якого показаний у лістингу 3.9.

Лістинг 3.9 — Реалізація перевірки специфікаціїї обов’язкових параметрів методу isset

```
if ((!isset(req.body.track, req.body.item, req.body.filter))
{
        res.status(400);
        res.json(
        {
                err: ‘Missing parameters’,
                msg:‘Required parameters are missing: “track”, “item”, “filter”.’,
        });
        return;
}
```

Під час завантаження файлу запити обробляються по-різному. Файли, які потрібно завантажити, також знаходяться всередині запитів як параметр. Вони не у форматі JSON, а в мультичасткових/форма-даних. Немає необхідності маніпулювати даними під час запису. Коли завантаження починається, вказується файл, у який буде записано, а після завершення завантаження сервер повертає відповідь, що операція була успішною. Для цього використовується розширення busboy. Коли запит отримано, він перевіряється, чи містить він необхідні дані, а потім дані перенаправляються у файл.

Щоб контролер працював належним чином, йому потрібен доступ до файлів у папці WORKER. Модуль File System (fs) дозволяє працювати з файловою системою. Використання функцій модуля fs подібне до функцій POSIX мови C. Усі функції мають синхронний та асинхронний варіанти. Настійно рекомендується використовувати асинхронну файлову систему для операцій введення-виводу. Останнім параметром асинхронних функцій є зворотний виклик, який викликається після завершення операції. Об’єкт зберігання помилок передається зворотному виклику як перший параметр, а

потім, за бажанням, дані операції. Зворотний виклик викликається в разі успіху та невдачі, промову вказує об’єкт 1-го параметра. Іншим модулем, пов'язаним з файловою системою, є path, який надає інструменти для роботи з файлами та шляхами до каталогів. У лістингу 3.10 показано, як виглядає читання асинхронного файлу.

Лістинг 3.10 — Приклад реалізації читання асинхронного файлу

```
fs.open(path.join(projectPath, ’processing’)), ’wx’, (err, fd) => {
        if(err) throw err;
        fs.close(fd, (err) => {
                if(err) console.error(err.stack);
        };
});
console.log(‘You can still wait for the file to load’);
```

Викликається функція fs.open, створює запит до файлової системи, і замість активного очікування функція переходить у сплячий режим і продовжує виконувати команди під fs.open.

У цьому випадку в терміналі буде відображатися текст «You can still wait for the file to load», і механізм JavaScript повернеться до роботи з файлом лише після того, як потрібний файл стане доступним. В результаті операції введення-виводу одного запиту не перешкоджають виконанню одночасних запитів.

Якщо під час належної обробки запиту виникає виняток, або якщо викликається наступна функція, маршрутизатор передає errorController обробнику запиту. Цей контролер містить лише функцію за замовчуванням. Помилка реєструється, і надсилається відповідь з кодом помилки 500 і відповідь JSON «Internal server error occured». Приклад коду наведено в лістингу 3.11.

Лістинг 3.11 — Приклад реалізації винятку помилки при обробці запиту

```
exports.default = (err, req, res, next) => {
        console.error(err.stack);
        res.status(500);
        res.json({
                msg: ’Internal server error occured.’,
        });
};
```

MainController обробляє запити, для яких HTML-відповідь, як очікується, буде відтворена веб-браузером. Використовується в 3 ситуаціях:

1) користувач відвідує вступну сторінку з можливістю створити новий проект;

2) користувач відображає сторінку з редактором;

3) користувач відображає сторінку з отриманим відео / з екраном, що повідомляє про стан обробки.

Робота з контролером проста, функція виклику якого представлена у лістингу 3.12. Коли відображається домашня сторінка або сторінка проекту, контролер передає лише запит на відтворення перегляду

Лістинг 3.12 — Прикдад виклику контролера для відображення сторінки

```
exports.main = (req, res) => res.render('main', {});
```

Запитуючи результуюче відео, контролер дивиться в каталозі проекту після прапорця обробки в процесі (обробка файлу). Якщо обробка триває, у поданні буде запропоновано відобразити екран очікування. Якщо відео не оброблено, воно побачить, чи існує отриманий файл output.mp4. Якщо файл не існує, проект не має результуючого відео, і повертається помилка 404. Якщо файл доступний, сам файл надсилається безпосередньо у відповіді, а не на сторінці HTML. Приклад реалізації методу доступу до файлової системи наведений у лістингу 3.13.

Лістинг 3.13 — Приклад реалізації методу доступу до файлової системи

```
fs.access(path.join(projectPath, req.params.projectID, 'processing'),
        fs.constants.R_OK, (err) => {
    if(err){
        const outputFile = path.resolve(path.join(projectPath,
                    req.params.projectID, 'output.mp4'));
        fs.access(outputFile, fs.constants.R_OK, (err) => {
            if (err) res.sendStatus(404);
            else res.sendFile(outputFile);
        });
    }
});
```

3.3 Розробка функціоналу рекордера екрану

В даному підрозділі розглянемо основні моменти у розробці додатку рекордера екрану. Він базується на архітектурі, що складається з клієнтської та серверної частин.

3.3.1 Розробка функціоналу клієнтської частини

Для клієнтської частини нам необхідно провести імпорт хуків, індикаторів завантаження, клієнт Socket.io і також потрібні стилі. Аби у нас був конкретний зв’язок із сервером то знадобиться нам створити декілька змінних:

— для адреси сервера;

— екземпляру типу MediaRecorder — інтерфейс, що надається MediaStream Recording API, для запису медіа;

— частин записаних даних;

— для генерації випадкового імені користувача (наприклад, User_1234), що буде, скоріш за все, вилучатись із об’єкту user, що міститься в контексті, наприклад, const {user} = useAuthContext(); const{username} = user;

— для виклику io(url, options), який повертає унікальний сокет клієнта, що використовується для передачі та отримання даних від сервера, де нам достатньо вказати адресу сервера.

— посилання на DOM-елементи;

— video для надання користувачеві можливості перегляду запису та її скачування.

Перше, що нам потрібно зробити на клієнті — це повідомити сервер про підключення нового користувача, повідомивши ім'я користувача, код якого приведений у лістингу 3.14.

Лістинг 3.14 — Приклад реалізації повідомлення сервера на нове підключення

```
useEffect(() => {
        socketRef.current.emit('user:connected', username.current)}, [])
```

Для надсилання подій використовується метод socket.emit(type, data), де type — рядок, що позначає тип події, а data — дані. Даними можуть бути як примітиви, і об'єкти. Для обробки подій використовується метод socket.on(type, callback), де type — Тип події, а callback — функція обробки, що приймає дані, надіслані за допомогою socket.emit.

Далі нам необхідно захопити екран (отримати потік відео), а також аудіодані з мікрофону користувача, приклад реалізації є представлений у лістингу 3.16.

Лістинг 3.15 — Приклад реалізації отримання аудіо та відео потоку

```
useEffect(() => {
        ;(async () =>{
                if (navigator.mediaDevices.getUserMedia) {
                        if (screenStream) {
                                try{
                                        const _voiceStream = await
                                        navigator.mediaDevices.getUserMedia({
                                        audio: true })
                                }
                        }
                }
        })()
 }, [screenStream])
```

Для отримання відео з захоплення екрана використовується метод getDisplayMedia(), що надається інтерфейсом MediaDevices, що входить до складу Navigator. На жаль, на сьогоднішній день даний метод підтримується тільки десктопними браузерами, що становить близько 35% користувачів , але це все ж таки краще, ніж нічого. Слід зазначити, що метож getDisplayMedia також вміє захоплювати аудіодані, але в даний час цю можливість підтримують тільки Edge і Chrome тому ми скористаємося іншим інтерфейсом. Для отримання аудіо-потоку використається метод getUserMedia, охват користувачів якого становить близько 93%.

Якщо спробуємо отримати потоки одночасно, то отримаємо тільки потоки відео, а спроба отримання аудіопотоку завершиться помилкою — Permission

denied. Принаймні така поведінка спостерігається в Chrome. Однак якщо виникне потреба у записі екрану без звуку, тому при виникненні будь-якої помилки, пов'язаної з отриманням аудіопотоку (включаючи відмову користувача у наданні дозволу на використання мікрофона), ми встановлюємо voiceStream значення unavailable таким чином — setVoiceStream('unavailable'), далі формуємо медіа-потік, приклад коду якого наведено у лістингу 3.16.

Лістинг 3.16 — Приклад реалізації формування медіа-потоку

```
let mediaStream
if (voiceStream === 'unavailable') {
        mediaStream = screenStream
} else
{
        mediaStream = new MediaStream([
        //…])
}
```

Склад медіа-потоку залежить від доступності аудіопотоку. Якщо аудіопотік недоступний, медіаєпоток складатиметься лише з відеопотоку. Інакше формується об'єднаний потік із відеотреків відеопотоку та аудіотреків аудіопотоку. Для об'єднання потоків використовується інтерфейс MediaStream.

3.3.2 Розробка функціоналу серверної частини

Першочергово, для сервера нам потрібно обов’язково створити екземпляри додатку Express, сервери на основі HTTP-запитів, та Socket.io. І особливо потрібно в тілі коду останнього вказати налаштування cors (з англ. Cross-Origin Resource Sharing) — заголовок, які сервер посилає браузеру при AJAX-запиті. Приклад наведено у лістингу 3.17. Він потрібен для запобігання несанкціонованого доступу до даних користувача, які знаходяться на сервері.

Лістинг 3.17 — Приклад створення Socket.io із вказанням CORS

```
const io = new Server(server, {
        cors: {
                origin: ‘ /*link*/’}
})
```

Далі потрібно зареєструвати обробку підключення шляхом імпорту функції для збереження запису, пошукової таблиці з id сокета — ім’я користувача, та частин даних. Приклад створення вказано у лістингу 3.18.

Лістинг 3.18 — Приклад виклику функції реєстратора обробки підключення

```
export const onConnection = (socket) =>{
        //TODO
}
```

Дана функція буде приймати за параметр — сокет. Саме підключення нового користувача буде оброблятись за допомогою запису імені користувача в пошукову таблицю. Після запису, кадри будуть передаватись на сервер.

Однак не виключаємо випадок, коли вкладка браузера буде випадково закрита під час запису — у такому випадку подія закінчення запису screenData:end відправлена не буде, і в найгіршому випадку втратимо 250 мс відео — mediaRecorder.start(250).

Для збереження файлу використаємо імпортовані утиліти з Node.js, аби створювати тимчасовий відеофайл по технології ffmpeg. Але для її роботи потрібна наявність локально встановленої даної утиліти. Вона, надаючи шлях до ffmpeg, преледає її також для обгортки ffmpeg-fluent.

Формат для тимчасового файлу обремо WebM, так як він є напрочуд легкий для відео, але близьким конкурентом для нього є MP4.

Для конвертації за допомогою ffmpeg потрібно:

— передати ffmpeg шлях до тимчасового файла;

— встановити налаштування конвертації;

— передати методу шлях для збереження конвертованого файлу, що відведено для тимчасових файлів;

— передати методу шлях до директорії, що відведено для тимчасових файлів

— після конвертації видалити тимчасовий файл.

Налаштування конвертації:

— -c — це кодек;

— -c:v — кодек для відео;

— -c:a — кодек для аудіо;

— -vf scale=1920:1080 — роздільна здатність відеофайлу, де як правило, чим менший параметр, тим менший розмір файлу і гірша якість відео;

— -crf 35 -b:v 0 — налаштування постійного коефіцієнт шивдкості та бітрейту відео.

Як кодек для відео ми вказуємо libvpx-vp9(VP9). Це пов'язано з тим, що для створення файлу формату WebM потрібно, щоб відео містилося в контейнері VP8 або VP9, а аудіо — Opus або Vorbis. Аудіо-доріжку ми просто копіюємо, вказуючи кодек.

Швидкість конвертації сильно залежить від обчислювальних потужностей, які ми маємо. У середньому час конвертації дорівнює тривалості відео. Конвертований файл виходить приблизно в 3 рази менше від оригіналу.

ВИСНОВКИ

Під час виконання бакалаврської дипломної роботи були досліджено та розроблено веб-додаток, що дозволяє записувати екран і також веб-відеоредактор.

У першому розділі охарактеризовано предметну область — здійснено порівняння веб-додатку і веб-сайту. Проаналізовано поняття веб-сервера, описано поняття веб-застосунка і веб-сайту, також наведено переваги та недоліки веб-сайтів. Також було проаналізовано сучасні інструменти для запису екрану і редагування відео із зазначенням їх особливостей, переваг, недоліків.

У другому розділі було здійснено розробку рішення: розглянуто теоретичні аспекти технологій, які використовувались в роботі, розроблено дизайн інтерфейсу користувача, а також розроблено API для відеоредактора.

У третьому розділі було проведено огляд структури проекту, розглянуто основні моменти для розробки функціоналу субдодатків, відповідно до їх архітектур.

На заключення, при розробці цієї дипломної роботи було створено працездатний веб-додаток, що містить у собі 2 веб-додатки, що можна застосовувати як продукт для створення медіаконтенту в розважальних, навчальних та корпоративних цілей. Без винятку, даний проект буде мати подальшу підтримку із метою розширення функціоналу, дизайну, а також сприяти інтеграції з іншими сервісами, соціальними мережами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. What is a web server?[Електронний ресурс] — режим доступу: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server

2. Що таке веб-сервер і для чого він [Електронний ресурс] — режим доступу: https://introserv.eu/ua/blog/hosting/scho-take-veb-server-%D1%96-dlya-chogo-v%D1%96n-potr%D1%96bnij/

3. Що таке веб додаток? [Електронний ресурс] — режим доступу: https://webcase.com.ua/uk/blog/cho-takoe-web-prilozhenie-vse-vidy/

4. Особливості веб-додатків [Електронний ресурс] — режим доступу: http://sites.znu.edu.ua/webprog/lect/1191.ukr.html

5. Вебзастосунок [Електронний ресурс] — режим доступу: https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA

6. Статичні та динамічні web-сайти [Електронний ресурс] — режим доступу: https://armedsoft.com/ua/blog/statychni-ta-dynamichni-web-sayty#:~:text=%D0%9F%D1%96%D0%B4%20%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%87%D0%BD%D0%B8%D0%BC%20%D1%80%D0%BE%D0%B7%D1%83%D0%BC%D1%96%D1%94%D1%82%D1%8C%D1%81%D1%8F%20%D1%81%D0%B0%D0%B9%D1%82%2C%20%D1%8F%D0%BA%D0%B8%D0%B9,%D1%8F%D0%BA%D0%BE%D0%BC%D1%83%20%D0%B2%D0%BE%D0%BD%D0%B0%20%D0%B7%D0%B1%D0%B5%D1%80%D1%96%D0%B3%D0%B0%D1%94%D1%82%D1%8C%D1%81%D1%8F%20%D0%BD%D0%B0%20%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D1%96

7. Статичний сайт [Електронний ресурс] — режим доступу: https://strong-planet.com.ua/static_site.html

8. Динамічний сайт [Електронний ресурс] — режим доступу: https://strong-planet.com.ua/dynamic_site.html

9. Різниця між статичним веб-сайтом та динамічним веб-сайтом [Електронний ресурс] — режим доступу:

https://ua.sawakinome.com/articles/internet/difference-between-static-website-and-dynamic-website-3.html

10. Відеоредактор [Електронний ресурс] — режим доступу: https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%B4%D0%B5%D0%BE%D1%80%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80

11. 5 дійсно безкоштовних відеоредакторів для Windows [Електронний ресурс] — режим доступу: https://habr.com/ru/post/318214/

12. Програми для запису відео з екрану комп'ютера. [Електронний ресурс] — режим доступу: https://www.imena.ua/blog/best-software-record-video-from-your-computer-screen/

13. Flash & The Future of Interactive Content [Електронний ресурс] — режим доступу: https://blog.adobe.com/en/publish/2017/07/25/adobe-flash-update#gs.ctytij

14. JavaScript Frameworks [Електронний ресурс] — режим доступу: https://www.wappalyzer.com/technologies/javascript-frameworks

15. Pasquali S., Faaborg K. Mastering Node.js: build robust and scalable real-time server-side web applications efficiently: підручник. Birmingham: Packt Publishing. №2, 2017. 498 с.

16. Nano ID Collision Calculator [Електронний ресурс] — режим доступу: https://zelark.github.io/nano-id-cc/

17. Tutorial: Intro to React [Електронний ресурс] — режим доступу: https://reactjs.org/tutorial/tutorial.html

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

проф., д.т.н.. Азаров О.Д..

" " 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання бакалаврської дипломної роботи

“ Веб-додаток з можливостями запису екранного робочого простору та подальшим редагуванням”

08-23.БДР.028.00.000 ТЗ

Науковий керівник: доцент к.т.н.

_____ Кожем’яко А. В.

Студент групи 2КІ-18б

_____ Підцерковний Є. О.

м. Вінниця – 2022

1 Підстава для використання бакалаврської кваліфікаційної роботи (БДР)

1.1 Актуальність розробки полягає у вирішенні потреб у медіаконтенті для медіаплатформ, а також через збільшення використання у корпоративному секторі дистанційної форми праці.

1.2 Наказ про затвердження теми бакалаврської дипломної роботи.

2 Мета і призначення БДР

2.1 Мета проекту — розробка зручного у використанні веб-додатку запису екрану та відеоредактора, який буде призначений для загального користування.

2.2 Призначення розробки — виконання бакалаврського дипломного проекту, аби його надалі впроваджувати і розвивати.

3 Вихідні дані для виконання БДР

3.1 Аналіз предметної області

3.2 Проведення аналізу сучасних проектів, що дозволяють записувати екран і редагувати відеоматеріали.

3.3 Розробка інтерфейсу та функціоналу веб-додатку.

4 Вимоги до виконання БДР

Головна вимога — розробити легкий та зручний веб-додаток.

5 Етапи БДР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

6 Матеріали, що подаються до захисту БДР

До захисту подаються: пояснювальна записка БДР, ілюстративні матеріали, протокол попереднього захисту БДР на кафедрі, відгук наукового керівника, анотації до БДР українською та іноземною мовами, довідка про відповідність оформлення БДР діючим вимогам.

Таблиця А.1 — Етапи БДР

№ з/п	Назва етапів виконання комплексної бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Постановка задачі роботи	08.03.22	розділ 1
2	Характеристика предметної області	09.03-14.03.22	розділ 1
3	Аналіз сучасних проектів відеоредакторів та записувачів екрану	17.03-01.04.22	розділ 1
4	Пошук необхідних технологій для розробки	03.04-09.04.22	розділ 2
5	Розробка веб-дизайну	12.04-16.04.22	розділ 2
6	Проектування API веб-додатку	19.04-27.04.22	розділ 2
7	Розробка функціоналу веб-додатку	01.05-18.05.22	розділ 3
8	Аналіз виконання роботи, висновки, додатки	21.05-25.05.22	Пояснювальна записка
9	Перевірка якості виконання бакалаврського проекту та усунення недоліків	28.05-09.06.22	Пояснювальна записка і презентація

7 Порядок контролю виконання та захисту БДР

Виконання етапів графічної та розрахункової документації БДР контролюється науковим керівником згідно зі встановленими термінами. Захист БДР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання БДР

При оформлюванні БДР використовуються:

— ДСТУ 3008 : 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302 : 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104-2006 «Едина система конструкторської документації. Основні написи»;

— документами на які посилаються у вище вказаних.

ДОДАТОК Б

Узагальнена структурна схема веб-додатку

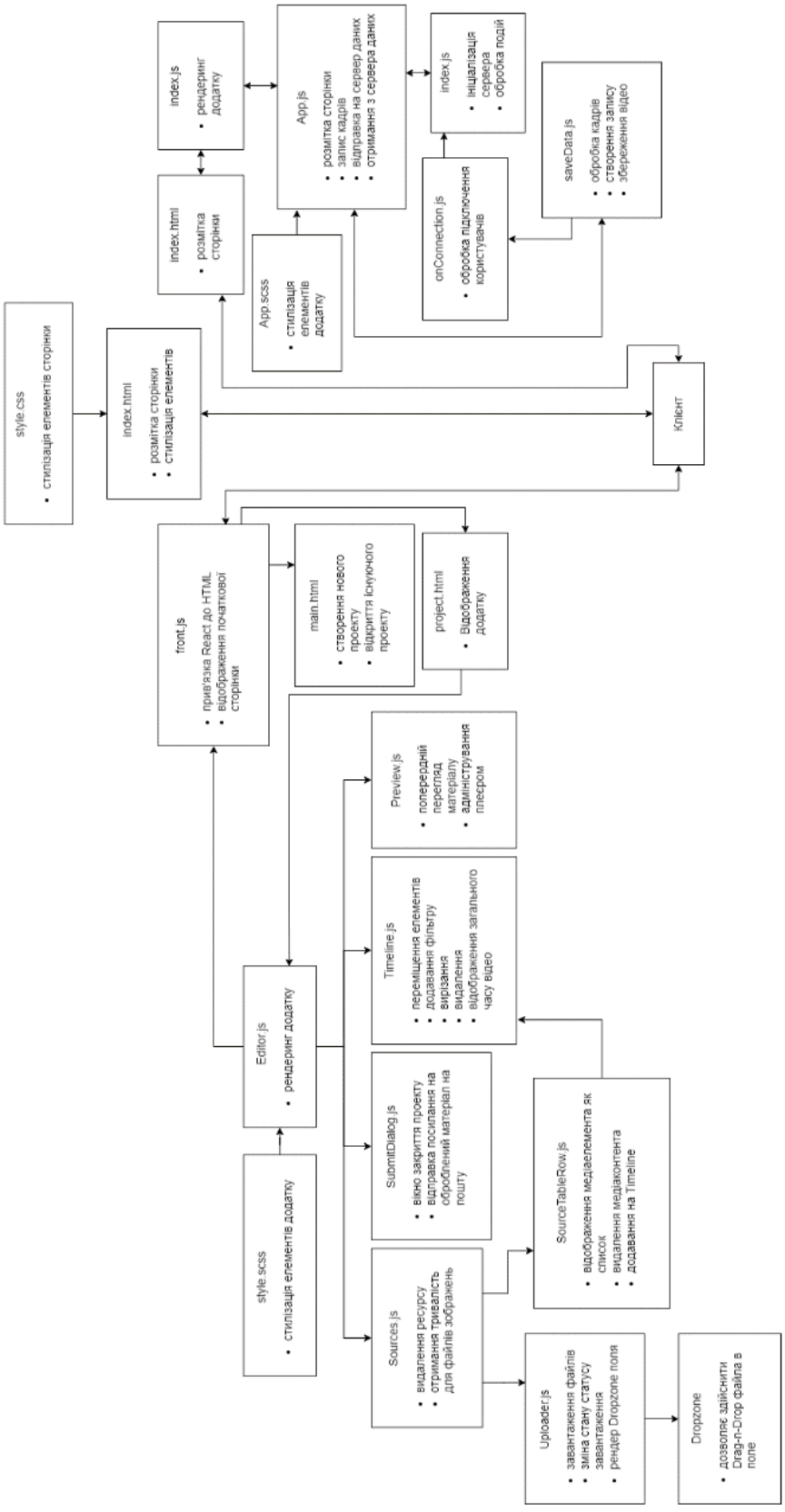

Рисунок Б.1 — Узагальнена структурна схема веб-додатку

ДОДАТОК В

Лістинг HTML файлу початкової сторінки

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Screen Recorder & Editor</title>
<link rel="icon" href="assets/img/favicon.png">
<link rel="stylesheet" href="assets/css/style.css">
</head>
<body>
<div class="jumbotron">
<div class="container-fluid d-flex flex-column align-items-center justify-content-around min-vh-100">
<div class="logo_text text-center">
<h1 class="display-1">Screen Recorder App</h1>
<h3>with video editor</h3>
</div>
<div class="buttons mb-5">
<div class="btn-group">
<a href="http://localhost:3000/">
<button type="button" class="menu-btn d-flex align-items-center flex-column p-5">
<img src="assets/img/screen_record.png" class="img1_btn" alt="screen recorder">
Start recording</button>
</a>
</div>
<div class="btn-group">
```

```
<a href="http://localhost:8080/">
<button type="button" class="menu-btn edit-btn d-flex align-items-center flex-column p-5">
<img src="assets/img/video_editor.png" class="img2_btn" alt="">Edit a video</button>
</a>
</div>
</div>
<footer>
Copyright &copy;2022 Eugene Pidtserkovnyi. All rights reserved.
</footer>
</div>
</div>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p" crossorigin="anonymous"></script>
</body>
</html>
```

ДОДАТОК Г

Лістинг CSS файлу початкової сторінки

```
@import url('https://fonts.googleapis.com/css2?family=Dancing+Script:wght@400;5
00;600;700&family=Pacifico&display=swap');
*{
    margin: 0;
    color: #0b2d39;
}
body{
    background-color: #4158D0;
    background-image: linear-gradient(43deg, #4158D0 0%, #C850C0 46%,
#FFCC70 100%);
}
.logo_text{
    font-family: 'Pacifico', cursive;
    font-weight: 700;
}
.menu-btn{
    width: 300px;
    padding: 5rem 6rem;
    background: rgba(255, 255, 255, 0.23);
    border-radius: 16px;
    box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);
    -webkit-backdrop-filter: blur(10.3px);
    backdrop-filter: blur(10.3px);
    border: 1px solid rgba(255, 255, 255, 0.23);
    margin: 1rem;
    font-family: 'Dancing Script', cursive;
    font-size: 25px;
```

```
}
.edit-btn{
    padding: 2.rem 6.7045rem;
}
.img1_btn{
    margin-left: 0.5rem;
}
.img1_btn, .img2_btn{
    margin-bottom: 1rem;
}
img{
    width: 100%;
}
footer{
    font-size: 0.85rem;
}
a, a:link, a:visited, a:active{
    text-decoration: none;
}
button:hover{
    background: rgba(255, 255, 255, 0.43);
    border: 1px solid rgba(255, 255, 255, 0.43);
    transition: .5s;
}
```

ДОДАТОК Д

Лістинг файлу клієнтської частини рекордера екрану

```
import { useEffect, useRef, useState } from 'react'
import Loader from 'react-loader-spinner'
import { io } from 'socket.io-client'
import './App.scss'
const SERVER_URI = 'http://localhost:4000'
let mediaRecorder = null
let dataChunks = []
function App() {
const username = useRef(`User_${Date.now().toString().slice(-4)}`)
const socketRef = useRef(io(SERVER_URI))
const videoRef = useRef()
const linkRef = useRef()
const [screenStream, setScreenStream] = useState()
const [voiceStream, setVoiceStream] = useState()
const [recording, setRecording] = useState(false)
const [loading, setLoading] = useState(true)
useEffect(() => {
    socketRef.current.emit('user:connect', username.current)
}, [])
useEffect(() => {
   ;(async () => {
      if (navigator.mediaDevices.getDisplayMedia) {
        try {
          const _screenStream = await navigator.mediaDevices.getDisplayMedia({
video: true
          })
          setScreenStream(_screenStream)
```

```
      } catch (e) {
        console.error('*** getDisplayMedia', e)
        setLoading(false)
      }
    } else {
      console.warn('*** getDisplayMedia not supported')
      setLoading(false)}
  })()
}, [])
useEffect(() => {
  ;(async () => {
    if (navigator.mediaDevices.getUserMedia) {
      if (screenStream) {
        try {
          const _voiceStream = await navigator.mediaDevices.getUserMedia({
            audio: true
          })
          setVoiceStream(_voiceStream)
        } catch (e) {
          console.error('*** getUserMedia', e)
          setVoiceStream('unavailable')
        } finally {
          setLoading(false)
        }}
    } else {
      console.warn('*** getUserMedia not supported')
      setLoading(false)
    }
  })()
}, [screenStream])
```

```
function startRecording() {
  if (screenStream && voiceStream && !mediaRecorder) {
    setRecording(true)
    videoRef.current.removeAttribute('src')
    linkRef.current.removeAttribute('href')
    linkRef.current.removeAttribute('download')
    let mediaStream
    if (voiceStream === 'unavailable') {
      mediaStream = screenStream
    } else {
      mediaStream = new MediaStream([
        ...screenStream.getVideoTracks(),
        ...voiceStream.getAudioTracks()
      ])
    }
    mediaRecorder = new MediaRecorder(mediaStream)
    mediaRecorder.ondataavailable = ({ data }) => {
      dataChunks.push(data)
      socketRef.current.emit('screenData:start', {
        username: username.current,
        data
      })}
    mediaRecorder.onstop = stopRecording
    mediaRecorder.start(250)
  }
}
function stopRecording() {
  setRecording(false)
  socketRef.current.emit('screenData:end', username.current)
  const videoBlob = new Blob(dataChunks, {
```

```
      type: 'video/webm'
    })
    const videoSrc = URL.createObjectURL(videoBlob)
    videoRef.current.src = videoSrc
    linkRef.current.href = videoSrc
    linkRef.current.download = `${Date.now()}-${username.current}.webm`
    mediaRecorder = null
    dataChunks = []
  }
  const onClick = () => {
    if (!recording) {
      startRecording()
    } else {
      if (mediaRecorder) {
        mediaRecorder.stop()
      }
    }}
  if (loading) return <Loader type='Oval' width='60' color='#0275d8' />
  return (
    <>
      <h1>Screen Recording App</h1>
      <video controls ref={videoRef}></video>
      <a ref={linkRef}>Download</a>
      <button onClick={onClick} disabled={!voiceStream}>
        {!recording ? 'Start' : 'Stop'}
      </button>
    </>
  )
}
export default App
```

ДОДАТОК Е

Лістинг файлу серверної частини рекордера екрану

```
import express from 'express'
import http from 'http'
import { Server } from 'socket.io'
import { onConnection } from './socket_io/onConnection.js'

const app = express()
const server = http.createServer(app)
const io = new Server(server, {
  cors: {
    origin: 'http://localhost:3000'
  }
})

io.on('connection', onConnection)

server.listen(4000, () => {
  console.log('Server ready 🚀 ')
})
```

ДОДАТОК Ж

Лістинг файлу стартової сторінки відеоредактора

```
import React, { Component } from 'react';
import Modal from 'react-modal';
import {server} from '../../config';
import FetchErrorDialog from '../editor/FetchErrorDialog';
Modal.setAppElement(document.body);
export default class NewProjectDialog extends Component {
	constructor(props) {
		super(props);
		this.state = {
			showFetchError: false,
			fetchError: '',
		};
		this.closeFetchErrorDialog = this.closeFetchErrorDialog.bind(this);
	}
	createProject() {
		const url = `${server.apiUrl}/project`;
		const params = {
			method: 'POST',
		};
		fetch(url, params)
			.then(response => response.json())
			.then(data => {
				if (typeof data.err === 'undefined') {
					window.location =
`${server.serverUrl}/project/${data.project}`;
				}
				else {
```

```
                        alert(`${data.err}\n\n${data.msg}`);
                    }
                })
                .catch(error => this.openFetchErrorDialog(error.message))
        ;
}
/**
 * Show Connection error dialog
 *
 * @param {String} msg
 */
openFetchErrorDialog(msg) {
        this.setState({
                showFetchError: true,
                fetchError: msg,
        });
}
/**
 * Close Connection error dialog
 */
closeFetchErrorDialog() {
        this.setState({
                showFetchError: false,
                fetchError: '',
        });
}
render() {
        return (
                <div>
```

```
                    {this.state.showFetchError && <FetchErrorDialog
msg={this.state.fetchError} onClose={this.closeFetchErrorDialog}/>}
                    <Modal
                        isOpen={true}
                        contentLabel="New project"
                        className={'modal'}
                        overlayClassName={'null'}
                    >
                        <h2 className={'logo'}><img
src={'/icons/favicon.svg'} alt={'logo'}/>Videoeditor</h2>
                        <div>
                            <button onClick={() =>
this.createProject()}>Create new project</button>
                        </div>
                    </Modal>
                </div>
            );
        }
}
```

ДОДАТОК К

Лістинг файлу mainController

```
import {config} from '../config';
const fs = require('fs');
const path = require('path');
exports.main = (req, res) => res.render('main', {});
exports.project = (req, res) => res.render('project', {});
exports.finished = (req, res) => {
	const outputFile = path.resolve(path.join(config.projectPath,
req.params.projectID, 'output.mp4'));
	fs.access(outputFile, fs.constants.R_OK, (err) => {
		if (err) {
			res.sendStatus(404);
		}
		else res.sendFile(outputFile);
	});
};
```

ДОДАТОК Л

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Веб-додаток з можливостями запису екранного робочого простору та подальшим редагуванням.

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 92,3% Схожість 7,7%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- ☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- ☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Підцерковний Є. О,
(підпис) (прізвище, ініціали)

Керівник роботи _____ Кожем'яка А. В,
(підпис) (прізвище, ініціали)