Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

ПРОГРАМНО-АПАРАТНІ ЗАСОБИ ДЛЯ ПРОВЕДЕННЯ ІНТЕРАКТИВНИХ БАГАТОКАНАЛЬНИХ ВІДЕОТРАНСЛЯЦІЙ

Виконав: студент 2 курсу, групи 1КІ-23м
Спеціальності 123 – Комп'ютерна інженерія
_____ Куклій Д. В.
Керівник: к.т.н., доцент кафедри ОТ
_____ Савицька Л. А.
«___» _____ 2024 р.
Опонент: к.ф-м.н., доцент кафедри МБІС
_____ Шиян А. А.
«___» _____ 2024 р.

Допущено до захисту
Завідувач кафедри ОТ
Азаров О. Д.
«___» _____ 2024 р.

ВНТУ 2024

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Галузь знань – Інформаційні технології
Освітній рівень – магістр
Спеціальність – 123 Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
_____ Азаров О. Д.
«18» вересня 2024 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту **Куклію Данилу Вячеславовичу**

Тема роботи: «Програмно-апаратні засоби для проведення інтерактивних багатоканальних відеотрансляцій», керівник роботи: Савицька Людмила Анатоліївна, затверджені наказом вищого навчального закладу № 310 від 17.09.2024.

1 Строк подання студентом роботи: 18.12.2024 р.

2 Вихідні дані до роботи: призначення системи – проведення інтерактивних багатоканальних відеотрансляцій

3 Зміст розрахунково-пояснювальної записки: вступ, аналіз сучасного стану апаратних та програмних засобів для проведення інтерактивних багатоканальних відеотрансляцій, постановка задачі, опис інструментів розробки, реалізація програми, економічна частина

4 Перелік графічного матеріалу: використано 7 таблиць та 7 рисунків.

6 Консультанти розділів роботи приведені у таблиці 1.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		видав	прийняв
1-4	Савицька Людмила Анатоліївна, к.т.н, доц.		
5	Небава Микола Іванович, к.е.н., проф.		

7 Дата видачі завдання 18.09.2024 р. Календарний план виконання МКР приведений у таблиці 2.

8 Календарний план виконання МКР приведений у таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів МКР	Строк виконання	Примітка
1	Постановка задачі	20.09.2024	Виконано
2	Огляд існуючих рішень	24.09.2024	Виконано
3	Теоретичні дослідження	29.09.2024	Виконано
4	Розробка структурної схеми	03.10.2024	Виконано
5	Розрахунок аналогової частини	07.10.2024	Виконано
6	Розробка принципової схеми	13.10.2024	Виконано
7	Моделювання роботи системи	20.10.2024	Виконано
8	Розрахунок економічної частини	27.10.2024	Виконано
9	Оформлення ПЗ та ілюстративного матеріалу	03.11.2024	Виконано
10	Виконання магістерської кваліфікаційної роботи	10.11.2024	Виконано
11	Перевірка якості виконання МКР та усунення недоліків	14.12.2024	Виконано
12	Підписи у керівника, опонента, нормоконтролера	15.12.2024	Виконано
13	Перевірка «антиплагіат»	17.11.2024	Виконано
14	Попередній захист	18.11.2024	Виконано

Студент Куклій Данило Вячеславович

Керівник Савицька Людмила Анатоліївна

АНОТАЦІЯ

УДК 004

Куклій Д. В. Програмне забезпечення для проведення інтерактивних багатоканальних трансляцій. Магістерська кваліфікаційна робота зі спеціальності 123 – Комп’ютерна інженерія. Вінниця: ВНТУ, 2024, 107 с.

Укр. мовою, Бібліогр.: 28 назв; рис.: 10 ; табл.: 7

У магістерській роботі здійснено огляд сучасних підходів до організації багатоканальних відеотрансляцій.

Проведено детальний аналіз та опис інструментів, що сприяють вирішенню поставленого завдання.

Розроблено технічну архітектуру та користувацький інтерфейс інтегрованої комп'ютерної системи.

Ключові слова: інтенгрована комп’ютерна система, C#, XAML, WPF, трансляція.

ABSTRACT

УДК 004

Kukliy D. V. Software for interactive multi-channel broadcasts. Master's qualification work in specialty 123 - Computer Engineering. Vinnytsia: VNTU, 2024, 107 p.

The master's work reviews modern approaches to organizing multi-channel video broadcasts.

A detailed analysis and description of the tools that contribute to solving the task is carried out.

The technical architecture and user interface of an integrated computer system are developed.

Keywords: integrated computer system, C#, XAML, WPF, broadcast.

ЗМІСТ

ВСТУП

Сучасний рівень технологічного розвитку людства розкриває широкі можливості, одна з яких – бачити, чути і спілкуватись на величезній відстані. А відеотрансляції стали невід'ємною частиною майже усіх масштабних подій, починаючи від різноманітних змагань, масових онлайн — конференцій, концертів і закінчуючи приватними стрімами або вебінарами. Такі трансляції забезпечують одночасну передачу медіаданих різного типу (відео, аудіо) з мінімальною затримкою, завдяки чому можна сказати, що усе відбувається наживо.

Актуальність дослідження визначається появою нових технологій, що відбувається постійно, відеотрансляції набувають нового рівня складності та функціональності, що вимагає удосконалення як програмної так і апаратної частин системи. Для підвищення якості й ефективності проведення відеотрансляцій необхідно розробити програмний продукт для зручної інтеграції різних медіапотоків в режимі реального часу. Причому важливо забезпечити потенційний продукт постійною підтримкою та оновленням, оскільки з часом будуть вдосконалюватись і виходити на ринок оновлені версії пристроїв вводу та виводу. Але для України в сучасних реаліях ця тема є особливо важливою й актуальною, оскільки відеотрансляції в режимі реального часу дають можливість демонструвати усьому світові події наживо, що привертає увагу світової спільноти до подій в Україні та дає можливість більш ефективно фіксувати воєнні злочини на території України.

Метою дослідження є створення програмного забезпечення, яке дасть можливість інтегрувати в єдину систему різні медіапотоки та пристрої, а також ефективно ними керувати в режимі реального часу [1].

Задачі дослідження:

— провести аналіз існуючих апаратних та програмних засобів, які дають можливість інтегрувати різні медіапотоки;

— розробити архітектуру програмного забезпечення для проведення багатоканальних інтерактивних відеотрансляцій;

— реалізувати функціонал програми, відповідно до технічного завдання;

— провести тестування та оптимізацію розробленого програмного продукту для забезпечення стабільної роботи в реальних умовах.

Об’єктом дослідження є технології та програмно-апаратні засоби для проведення багатоканальних відеотрансляцій.

Новизна одержаних результатів полягає у розробці програмного забезпечення, яке дає можливість синхронізувати потоки даних з різних апаратних частин в єдиній системі, що дає можливість проводити відеотрансляції на новому рівні.

Практичне значення одержаних результатів полягає в розробці та впровадженні програмного забезпечення для інтерактивних багатоканальних відеотрансляцій, що дозволяє ефективно управляти різними потоками даних (відео, аудіо, зображення) в реальному часі, забезпечуючи високу якість трансляцій та інтерактивність для користувачів у різних сферах, таких як освіта, бізнес, медіа та онлайн-майданчики. Це програмне рішення сприяє зниженню витрат часу та ресурсів на організацію трансляцій і забезпечує зручний інтерфейс для керування контентом.

Апробація результатів: Л. А. Савицька, Д. В. Куклій, Н. В. Добровольська. СТРІМІНГОВІ ПЛАТФОРМИ ТА ЇХ РОЛЬ У РОЗВИТКУ МУЛЬТИМЕДІЙНОЇ КОМУНІКАЦІЇ // [Електронний ресурс] / Л. А. Савицька // Матеріали LIV Всеукраїнської науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії, Вінниця, 24-27 березня 2025 р. – Електрон. текст. дані. – 2024. – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/paper/view/22729

1 АНАЛІЗ СУЧАСНОГО СТАНУ АПАРАТНИХ ЗАСОБІВ ДЛЯ ПРОВЕДЕННЯ ІНТЕРАКТИВНИХ БАГАТОКАНАЛЬНИХ ВІДЕОТРАНСЛЯЦІЙ

1.1 Огляд апаратних засобів для проведення відеотрансляцій

Огляд апаратних засобів для проведення відеотрансляцій охоплює комплексний аналіз обладнання, яке забезпечує передачу відео та аудіосигналів високої якості в реальному часі. До основних складових належать відеокамери, мікрофони, апаратні кодери та комутатори, що відіграють ключову роль у процесі захоплення, обробки та передачі медіаконтенту. Відеокамери забезпечують зйомку зображення у високій роздільній здатності (HD, 4K та вище), що є критично важливим для професійних трансляцій, таких як спортивні події, вебінари чи концерти. Мікрофони, у свою чергу, відповідають за захоплення чистого та чіткого звуку, що дозволяє уникати сторонніх шумів і забезпечує якісну аудіосупровід трансляцій.

Одним із ключових параметрів камери [2], що визначає якість відеозображення є роздільна здатність, оскільки вона вказує на кількість пікселів, з яких складається кадр. Чим більше пікселів містить зображення, тим деталізованішим, чіткішим і реалістичнішим воно буде виглядати. Сучасні камери підтримують різні стандарти роздільної здатності, що дозволяє обирати оптимальну якість відео залежно від вимог проєкту та технічних можливостей обладнання.

Одним із базових стандартів є HD (High Definition), що має роздільну здатність 1280x720 пікселів. Цей формат забезпечує достатньо чітке зображення для багатьох випадків використання і став мінімальним стандартом для сучасних відеотрансляцій. Однак із розвитком технологій більшість відеоконтенту сьогодні орієнтується на вищу якість, таку як Full HD, яка відповідає роздільній здатності 1920x1080 пікселів. Цей формат є найпоширенішим для споживчого контенту,

включаючи трансляції на платформах YouTube чи соціальних мережах, оскільки він забезпечує чіткі деталі при збереженні відносно помірного обсягу даних.

Наступним кроком у еволюції роздільних здатностей стало впровадження формату 2K з розміром кадру 2048x1080 пікселів. Цей формат часто використовується в кінематографічних проєктах, оскільки він дозволяє отримати відео з підвищеною деталізацією порівняно з Full HD, що особливо помітно на великих екранах. Подальший розвиток технологій призвів до появи роздільної здатності 4K UHD (Ultra High Definition) з параметрами 3840x2160 пікселів. 4K став новим стандартом для професійного відеовиробництва, трансляцій та кінематографа завдяки своїй здатності забезпечувати надзвичайно високу деталізацію зображення. Ця роздільна здатність дозволяє масштабувати та обрізати відео без втрати якості, що робить її особливо цінною для створення контенту, який буде переглядатися на великих екранах або пристроях із високою щільністю пікселів.

На піку сучасних технологій знаходиться формат 8K, що відповідає роздільній здатності 7680x4320 пікселів. Цей рівень якості забезпечує надзвичайно деталізоване зображення, яке перевершує будь-які попередні стандарти. Використання 8K є актуальним для високобюджетних проєктів, таких як кінематографічна зйомка чи трансляція масштабних подій. Однак робота з такими файлами потребує потужного апаратного забезпечення, оскільки обробка, передача та збереження даних у форматі 8K вимагає значних ресурсів.

Ще одним важливим параметром відеозйомки є частота кадрів [3], оскільки вона визначає кількість окремих зображень (кадрів), які камера фіксує за одну секунду. Ця характеристика вимірюється у FPS (frames per second), і чим вищим є показник частоти кадрів, тим плавнішим і реалістичнішим буде виглядати відео, особливо коли йдеться про сцени зі швидким рухом.

Одним із найпоширеніших стандартів є 24 FPS, який вважається класичним кінематографічним форматом. Ця частота кадрів використовується у художніх фільмах і створює характерний «кінематографічний» ефект, що відрізняється від інших форматів своєю легкою нерівномірністю рухів, яку глядач сприймає як звичний стиль кінематографу. Завдяки цьому відео у 24 FPS часто виглядає більш атмосферним і емоційно виразним.

Частота 30 FPS є стандартом для телевізійних трансляцій, а також широко використовується на популярних відеоплатформах. У порівнянні з 24 FPS, цей формат забезпечує дещо плавніший рух, що робить його ідеальним для більшості телевізійних шоу, новин та онлайн-контенту. Однак, відео, зняте з частотою 30 FPS, втрачає характерний кінематографічний ефект і більше підходить для практичного відображення реалістичних подій.

60 FPS забезпечує ще вищу плавність рухів, завдяки чому цей формат ідеально підходить для зйомки динамічних сцен, таких як спортивні події, екшен-сцени у фільмах або геймплей у відеоіграх. Ця частота кадрів також стала стандартом для відео високої якості на платформах на кшталт YouTube, оскільки вона дозволяє користувачам насолоджуватися чітким і плавним відео навіть під час швидких рухів або змін кадрів. В результаті 60 FPS стає популярним вибором для контенту, де важливо зберегти максимальну реалістичність і динамічність.

При зйомці з частотою 120 FPS і вище з’являється можливість створювати відео в режимі уповільненого відтворення (slow motion). Висока частота кадрів дозволяє значно сповільнювати рухи без втрати плавності і чіткості зображення. Це особливо актуально для зйомок унікальних або складних динамічних сцен, таких як спортивні повтори, кінематографічні спецефекти чи експериментальні відео. Чим вища частота кадрів при зйомці, тим більше варіантів для постобробки відео, що відкриває можливості для креативних і технічних рішень.

Автоматичне фокусування [4] відіграє критичну роль у підтриманні чіткого та детального зображення під час зйомки, особливо коли об’єкти постійно рухаються або змінюється відстань до них. Сучасні камери оснащені різними технологіями автофокусу, які забезпечують оптимальну якість зображення залежно від умов зйомки. Одним із найпоширеніших методів є контрастний автофокус, що працює на основі аналізу зображення та визначення ділянок з максимальним контрастом. Ця технологія є надзвичайно точною, але її недоліком є відносно низька швидкість, що робить її найбільш ефективною для статичних сцен та фотозйомки. У випадках, коли об’єкт перебуває в русі, використовується фазовий автофокус, який швидко визначає фокусну відстань завдяки спеціальним сенсорам. Ця технологія дозволяє миттєво фокусуватися на об'єктах, що робить її ідеальною для відеозйомки, особливо динамічних сцен, де важлива оперативність.

Для досягнення максимальної ефективності в умовах, що вимагають і точності, і швидкості, камери часто використовують гібридний автофокус, який поєднує переваги контрастного та фазового методів. Це дозволяє отримати чітке зображення навіть у складних умовах, таких як слабке освітлення чи швидка зміна положення об'єкта у кадрі. Особливо важливим є автофокус на очах, який став технологічним проривом у портретній зйомці. Ця функція дозволяє камері автоматично розпізнавати очі об’єкта та фокусуватися саме на них, що забезпечує високу деталізацію обличчя. У сучасних камерах система автофокусу на очах може працювати навіть при русі об’єкта, що робить її незамінною для динамічних портретів або відеозйомки людей.

Не менш важливою технологією, що підтримує якість зображення, є стабілізація зображення [5], яка компенсує тремтіння або випадкові рухи камери під час зйомки. Оптична стабілізація є найбільш ефективним методом, оскільки вона працює на апаратному рівні за допомогою рухомих елементів об’єктива чи сенсора. Цей метод мінімізує вплив механічних рухів і не погіршує якість

зображення. Альтернативою є електронна стабілізація, яка працює на програмному рівні, компенсуючи рух камери шляхом обрізки та корекції кадрів. Хоча цей метод може знизити роздільну здатність і кут огляду відео, він залишається популярним завдяки своїй гнучкості та доступності. Найбільш просунутим рішенням є гібридна стабілізація, що поєднує оптичні та електронні методи для досягнення максимальної стабільності у різних умовах зйомки.

Огляд сучасних засобів для аудіо в рамках проведення відеотрансляцій також є важливою частиною загальної системи забезпечення якісного контенту. Аудіо часто сприймається як доповнення до відео, проте на практиці саме якість звуку може стати ключовим фактором, що визначає успішність трансляції. Сучасні мікрофони та інші аудіопристрої, які використовуються для зйомок і трансляцій, мають низку технічних характеристик, що безпосередньо впливають на якість звуку, включаючи чутливість, частотний діапазон, напрямність і технології придушення шуму.

Огляд апаратних засобів для проведення аудіотрансляцій охоплює комплексний аналіз обладнання, що забезпечує захоплення, обробку та передачу звукового сигналу високої якості в реальному часі. До основних складових належать мікрофони, аудіоінтерфейси, мікшери та підсилювачі, кожен з яких відіграє важливу роль у забезпеченні чистого й чіткого звуку для трансляцій, концертів, вебінарів чи інших подій. Мікрофони є першочерговим елементом аудіосистеми, оскільки вони здійснюють початкове захоплення звукового сигналу.

Однією з головних характеристик мікрофонів є чутливість [6], що визначає, наскільки точно мікрофон здатен передати звукові хвилі. Високочутливі мікрофони забезпечують реалістичну передачу звуку з високою деталізацією, однак у той же час вони більш схильні до захоплення сторонніх шумів, що вимагає ретельного налаштування та використання шумознижувальних технологій.

Іншим важливим параметром є частотний діапазон, який визначає діапазон звукових частот, що може відтворити мікрофон. Для якісного аудіо професійні мікрофони підтримують широкий частотний діапазон, що охоплює як низькі, так і високі частоти, дозволяючи отримати повноцінне звучання голосу чи музичних інструментів.

Напрямність мікрофона [7] є ще одним ключовим параметром, який визначає, з якого напрямку мікрофон вловлює звук. Кардіоїдні мікрофони мають спрямовану діаграму чутливості і захоплюють звук переважно з переднього напрямку, що дозволяє ефективно ізолювати голос мовця від фонових шумів. Такі мікрофони є найпопулярнішими для трансляцій, інтерв'ю та студійного запису. Водночас всеспрямовані мікрофони захоплюють звук з усіх боків, що робить їх ідеальними для запису навколишнього середовища або групових виступів.

Суперкардіоїдні та гіперкардіоїдні мікрофони, які мають ще вужчу діаграму спрямованості, використовуються у випадках, коли необхідно ізолювати джерело звуку від усіх інших шумів. Для досягнення максимальної якості звуку важливими є технології придушення шумів, які дозволяють мінімізувати фонові шуми, вібрації та інші небажані звуки під час захоплення аудіо. Серед таких технологій варто відзначити активне шумозаглушення, яке працює на основі створення звукових хвиль, що нейтралізують шуми, а також поп-фільтри та вітрозахист, які використовуються для зменшення небажаних звукових артефактів під час мовлення.

Аудіоінтерфейси відіграють критичну роль у забезпеченні передачі звуку від мікрофонів до комп'ютерів або інших пристроїв для подальшої обробки та трансляції. Вони конвертують аналоговий сигнал, що надходить від мікрофонів, у цифровий, забезпечуючи мінімальні затримки та високу якість аудіо. Основними характеристиками аудіоінтерфейсів є розрядність та частота дискретизації, які визначають, наскільки точно пристрій оцифровує аналоговий сигнал. Професійні

аудіоінтерфейси зазвичай підтримують 24-бітну розрядність та частоту дискретизації до 192 кГц, що забезпечує високий рівень деталізації звуку. Крім того, аудіоінтерфейси оснащуються передпідсилювачами для мікрофонів, які підвищують рівень сигналу та зберігають його чистоту без додавання шумів або спотворень.

Мікшери [8] є ще одним важливим елементом системи аудіотрансляцій, оскільки вони дозволяють регулювати рівні сигналів від різних джерел, застосовувати ефекти та змішувати аудіо для створення єдиного звукового потоку. Цифрові мікшери пропонують розширені можливості, включаючи автоматизацію процесів, збереження налаштувань та обробку звуку в реальному часі. Вони також підтримують інтеграцію з програмним забезпеченням для обробки аудіо, що робить їх ідеальними для складних трансляцій та студійних проєктів. Аналогові мікшери, хоча й мають менше функціональних можливостей, забезпечують високу надійність і простоту у використанні, що робить їх популярними для невеликих заходів і живих виступів.

Підсилювачі використовуються для підвищення рівня аудіосигналу до необхідної потужності для передачі його на динаміки або системи моніторингу. Важливими характеристиками підсилювачів є потужність та коефіцієнт гармонійних спотворень, які визначають якість та чистоту відтвореного звуку. Сучасні підсилювачі забезпечують мінімальні спотворення і працюють ефективно навіть при високих навантаженнях, що робить їх незамінними для масштабних трансляцій та концертів.

Також важливою складовою аудіосистем є моніторинг, який дозволяє контролювати звук у реальному часі. Студійні монітори та навушники забезпечують точне відтворення аудіо без додаткового підсилення чи корекції частот, що дозволяє операторам і звукорежисерам оперативно налаштовувати звук для досягнення найкращого результату.

1.2 Огляд технологій для багатоканального показу

Для забезпечення ефективної передачі кількох потоків медіаданих одночасно необхідно використовувати відповідні стандарти і протоколи, що дозволяють організувати стабільний зв'язок та синхронізувати різні типи даних. Одним із найбільш поширених стандартів для передачі відео та аудіо у багатоканальних системах є MPEG-TS (MPEG Transport Stream) [9]. Цей стандарт забезпечує можливість об'єднувати декілька потоків, таких як відео, аудіо та супутні дані, в єдиний потік для ефективної передачі та їх подальшої синхронізації. MPEG-TS широко використовується у цифровому телебаченні, оскільки дозволяє передавати кілька каналів одночасно в межах одного транспортного потоку, забезпечуючи оптимальну організацію даних.

Ще одним важливим протоколом є RTMP (Real-Time Messaging Protocol), який забезпечує передачу аудіо, відео та інших даних у режимі реального часу через інтернет. Завдяки своїй здатності передавати кілька потоків одночасно, RTMP активно використовується на стрімінгових платформах, таких як YouTube та Twitch, де реальна швидкість передачі й мінімальна затримка є критично важливими. Поряд із RTMP для організації багатоканальних трансляцій часто застосовуються протоколи HLS (HTTP Live Streaming) і DASH (Dynamic Adaptive Streaming over HTTP), які використовують технологію адаптивного стрімінгу. Це дозволяє передавати медіадані різної якості й роздільної здатності, автоматично підлаштовуючись під змінну пропускну здатність мережі, що забезпечує стабільне й плавне відтворення відео навіть при поганому інтернет-з'єднанні.

Ключовим елементом багатоканального показу є забезпечення синхронізації потоків, оскільки для коректного відтворення необхідно гарантувати їх одночасне надходження. Для цього застосовуються стандарти, такі як NTP (Network Time Protocol) [10], який забезпечує синхронізацію часу між пристроями у мережі, та

PTP (Precision Time Protocol) [11], що дозволяє досягати більшої точності в синхронізації, особливо в умовах професійних трансляцій.

Технологія віртуальних відеоматриць або відео-стін є ще одним важливим аспектом багатоканального показу, оскільки дозволяє одночасно виводити кілька потоків на різні дисплеї. Такі системи часто використовуються в телевізійних студіях, на спортивних подіях чи концертах, де необхідно показати різні джерела відео, як от камери, презентації або відеофайли. Для управління цими потоками застосовується спеціалізоване програмне забезпечення, що дає змогу оператору контролювати й комбінувати різні джерела в реальному часі.

Ще одним важливим аспектом є використання багатоканальних стрімінгових платформ [12], таких як OBS Studio, Wirecast та vMix. Ці програми забезпечують можливість інтегрувати кілька потоків відео та аудіо з різних джерел, як от камери, мікрофони чи інші мультимедійні пристрої, і транслювати їх одночасно через інтернет. Такі платформи надають інструменти для синхронізації, редагування та змішування потоків у реальному часі, що дозволяє створювати високоякісні мультимедійні шоу.

Наприклад, vMix підтримує роботу з кількома камерами та пристроями, забезпечуючи гнучкість у налаштуванні й керуванні потоками під час великих трансляцій. Wirecast, у свою чергу, дає можливість записувати та синхронізувати кілька потоків контенту для подальшого редагування та відтворення, що робить його особливо популярним для трансляцій концертів, спортивних подій чи вебінарів.

Для обробки та передачі багатоканальних потоків використовуються не лише програмні, а й апаратні рішення. Сучасні мультимедійні процесори дозволяють одночасно обробляти кілька відеопотоків у високій якості, зберігаючи при цьому чіткість і деталізацію зображення. Відеомікшери та аудіомікшери забезпечують

можливість керувати різними джерелами сигналів, комбінувати їх у єдине відео або перемикатися між ними у реальному часі.

Окрім локального обладнання, важливу роль відіграють хмарні рішення, які дозволяють обробляти й транслювати потоки без необхідності використовувати дорогі сервери чи потужні комп'ютери. Хмарні платформи, як от AWS Elemental MediaLive або Google Cloud Media Services, забезпечують масштабованість системи й можливість інтеграції з іншими сервісами для організації стабільних багатоканальних трансляцій.

Організація багатоканального показу, попри всі його переваги, зіштовхується з певними викликами. Однією з основних проблем є забезпечення синхронізації різних потоків, оскільки будь-яке відставання або розсинхронізація між відео та аудіо може значно погіршити якість трансляції. Крім того, багатоканальна передача вимагає високої продуктивності системи, оскільки обробка відео у форматах 4K або 8K вимагає значних обчислювальних ресурсів. Це включає потужні процесори, графічні відеокарти та швидкісні накопичувачі для обробки великих обсягів даних у реальному часі.

Ще одним викликом є необхідність у стабільному й швидкому інтернет-з'єднанні, адже передача кількох потоків у високій якості вимагає значної пропускної здатності мережі. У разі нестабільного з'єднання можуть виникати проблеми, пов'язані з буферизацією, затримками або втратою кадрів, що негативно вплине на загальну якість трансляції.

Програмні засоби, які використовуються для проведення багатоканальних трансляцій та показу, є важливим компонентом сучасних медіасистем. Вони дозволяють не лише обробляти та транслювати кілька потоків одночасно, а й забезпечують інтеграцію різних типів медіаданих, таких як відео, аудіо, текст і графіка, для створення цілісного та якісного контенту. У цьому розділі буде проведено порівняльний аналіз кількох найпопулярніших програмних засобів, що

використовуються для багатоканальних трансляцій, з урахуванням їх функціональних можливостей, продуктивності, зручності використання та особливостей інтеграції з різними апаратними пристроями.

Одним із найбільш потужних та функціонально насичених програмних засобів для проведення інтерактивних багатоканальних відеотрансляцій є vMix. Ця платформа забезпечує широку підтримку як апаратних, так і програмних засобів, що дозволяє організовувати трансляції з різноманітними потоками даних, включаючи відео, аудіо, зображення та візуальні ефекти у режимі реального часу. Одна з ключових переваг vMix полягає у можливості інтеграції з популярними сервісами відеоконференцій, такими як Zoom, що відкриває нові можливості для проведення онлайн-заходів. Зокрема, vMix здатний приєднуватися до Zoom-зустрічі як окремий учасник та отримувати прямий доступ до відео та аудіопотоків усіх учасників конференції у роздільній здатності до 1080p HD. Це дозволяє гнучко перемикатися між камерами та екранами учасників, динамічно відображати як індивідуальних спікерів, так і весь залучений контент одночасно. Така функціональність особливо актуальна для заходів, які передбачають постійну комунікацію між учасниками і модераторами, а також інтерактивні панелі обговорень.

Ще одним суттєвим нововведенням у vMix є підтримка сучасних відеокодеків AV1 та HEVC для RTMP-стрімінгу. Ці формати забезпечують значно вищу якість відео при тому ж рівні бітрейту у порівнянні з традиційним H.264, що критично важливо при трансляції контенту високої роздільної здатності. Для повноцінного використання цих функцій потрібні сучасні графічні карти, такі як NVIDIA GeForce 4050 для AV1 або 2050 для HEVC. Завдяки цьому vMix стає інструментом, здатним підтримувати вимоги до якості трансляції навіть у найбільш ресурсомістких середовищах, таких як спортивні трансляції, музичні концерти чи наукові конференції.

Особливу увагу у vMix приділено гнучкій роботі зі сценами та багатошаровими відеовхідними потоками. Нова функція Layer Designer дозволяє точно налаштовувати позиціонування, розмір та обрізку шарів на екрані, що значно спрощує створення складних композицій з кількома потоками відео чи графічних елементів. Завдяки підтримці різних режимів редагування, включаючи переміщення, обрізку та налаштування рамок, користувач може досягти візуально привабливих результатів навіть без поглиблених знань графічного дизайну. До того ж, функція Undo/Redo у Layer Designer забезпечує можливість швидкого корегування помилок, що значно покращує зручність роботи.

Важливою складовою сучасного функціоналу vMix є можливість роботи з аудіопотоками завдяки новим інструментам, таким як Audio Bus Manager та Bus Mixer. Ці інструменти дозволяють гнучко керувати аудіо на рівні кожного окремого входу, налаштовуючи гучність для кожного потоку або створюючи індивідуальні мікси для різних каналів трансляції. Підтримка функції Pre-Fader Listen дозволяє здійснювати попередній моніторинг звуку, що є важливим для забезпечення високої якості аудіоконтенту під час трансляцій.

Однією з важливих новацій у vMix є підтримка SRT (Secure Reliable Transport) для вхідних і вихідних потоків. Це дозволяє використовувати надійні з'єднання для передачі відео навіть у випадках нестабільного інтернет-з'єднання. Крім того, SRT може бути використано як джерело для функції Instant Replay, що особливо актуально для спортивних трансляцій, де потрібен швидкий повтор ключових моментів.

Разом із тим, vMix надає широкі можливості для кастомізації інтерфейсу та автоматизації завдань завдяки понад 150 новим шорткатам та інтеграції з пристроями управління, такими як Stream Deck. Це дозволяє спрощувати роботу оператора під час складних багатоетапних трансляцій та підвищувати ефективність управління контентом.

Варто зазначити, що попри всі переваги vMix, він вимагає досить потужного апаратного забезпечення для забезпечення стабільної роботи, особливо при обробці 4K- та 8K-відео, або при використанні сучасних відеокодеків. Крім того, хоча інтерфейс програми є інтуїтивно зрозумілим для досвідчених користувачів, новачкам може знадобитися час на освоєння всіх функцій та налаштувань.

Інтерфейс цього багатофункціонального програмного забезпечення показано на рисунку 1.

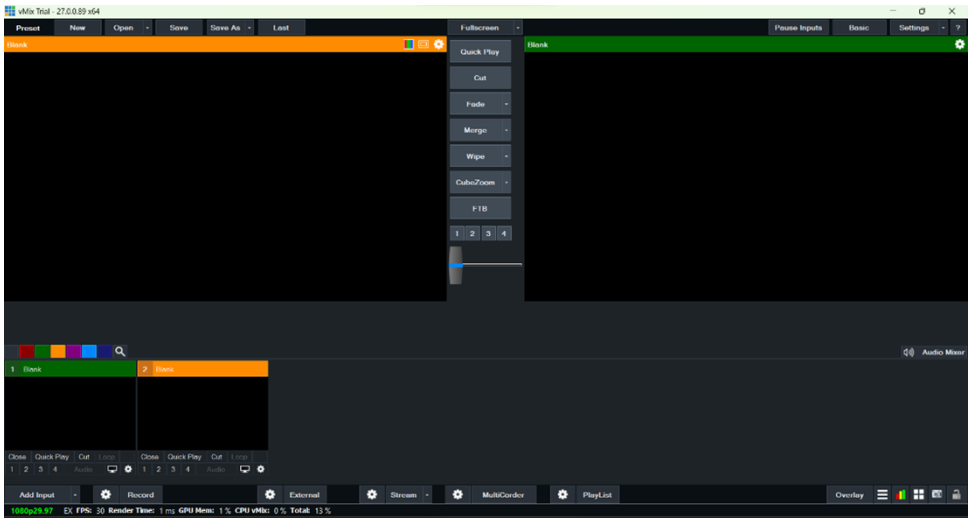

Рисунок 1 — Інтерфейс vMix

Іншим популярним інструментом для багатоканальних трансляцій є Wirecast. Wirecast є одним із найбільш універсальних і гнучких програмних інструментів для організації інтерактивних багатоканальних відеотрансляцій, що поєднує у собі функціонал професійної студії продакшну. Програмне забезпечення дозволяє перетворити звичайний ноутбук чи комп'ютер на повноцінний центр обробки відеоконтенту, надаючи користувачеві всі необхідні засоби для створення високоякісних трансляцій у реальному часі.

Однією з ключових особливостей Wirecast є підтримка необмеженої кількості джерел живого відео. Це включає роботу з підключеними камерами, IP-

камерами, мобільними пристроями та іншими потоковими джерелами. Завдяки інтеграції з PTZ-контролерами, оператори можуть дистанційно керувати камерами з можливістю панорамування, нахилу та масштабування, що робить Wirecast особливо зручним для великих подій, таких як конференції, концерти чи спортивні матчі.

Wirecast вирізняється своєю гнучкістю та розширеними можливостями для творчого налаштування трансляції. Програма використовує шарову композицію для роботи зі сценами, що дозволяє одночасно додавати та налаштовувати різноманітні візуальні елементи, такі як графічні накладення, анімації, текст та живі відеопотоки. Це дозволяє створювати динамічний та професійний вигляд контенту, максимально адаптуючи його під специфічні потреби аудиторії. Крім того, Wirecast підтримує функцію ISO-запису, яка дозволяє здійснювати незалежний запис кожного вхідного джерела відео у високій якості. Це особливо корисно для подальшого редагування відеоматеріалу після завершення трансляції. Віртуальні камери та мікрофони, що є частиною функціоналу Wirecast, дозволяють передавати відео- та аудіопотоки у сторонні програми, такі як Zoom чи Microsoft Teams, що робить інструмент універсальним для гібридних подій та вебінарів.

Ще однією важливою перевагою Wirecast є його вбудована підтримка багатоплатформенного стрімінгу. Програма дозволяє здійснювати одночасну трансляцію на декілька платформ, таких як YouTube, Facebook, а також RTMP-сервери, використовуючи готові пресети. Це значно спрощує процес налаштування та дозволяє ефективно охопити широку аудиторію без додаткових ресурсів. Важливою складовою є підтримка хромакею, яка дозволяє користувачам замінювати фон у реальному часі, що особливо актуально для інтерв'ю, вебінарів чи новинних трансляцій. Анімаційні графічні елементи та оверлеї додають професійного вигляду стріму, забезпечуючи його візуальну привабливість та динамічність.

Wirecast також пропонує унікальну медіатеку з понад 500,000 попередньо завантажених графічних та відеоресурсів, що значно полегшує процес створення візуального контенту. Користувачі можуть обирати з готових анімацій, заставок, відеокліпів та аудіофайлів для швидкого та зручного оформлення трансляції без потреби у зовнішньому редагуванні чи залученні додаткових дизайнерів. Такий підхід дозволяє значно скоротити час підготовки до ефіру, забезпечуючи при цьому високу якість кінцевого продукту.

Проте варто зазначити, що для стабільної роботи Wirecast у випадку обробки кількох потоків у високій роздільній здатності необхідне потужне апаратне забезпечення. Використання функцій ISO-запису, багатопотокового стрімінгу та віртуальних камер вимагає від системи значних обчислювальних ресурсів, особливо якщо трансляція ведеться у форматі Full HD або 4K.

Інтерфейс Wirecast показано на рисунку 2.

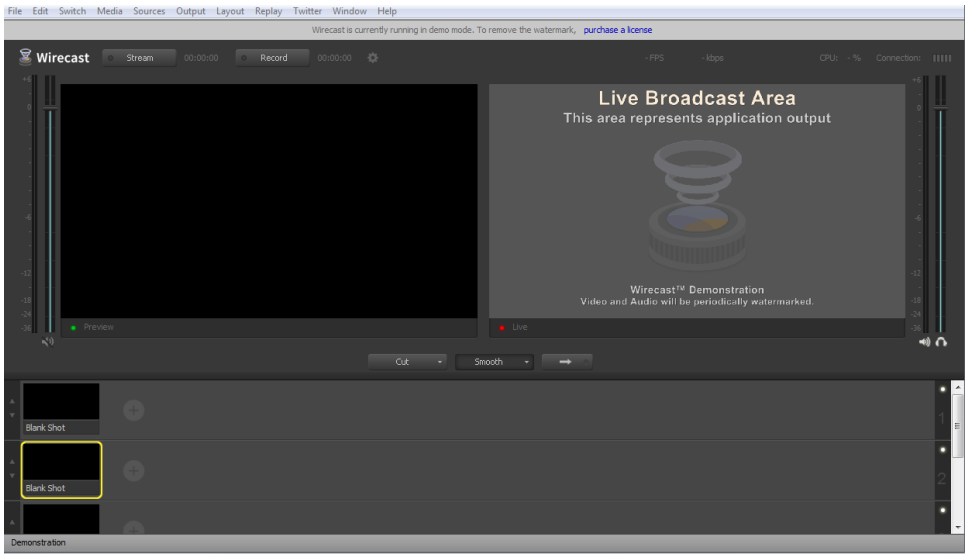

Рисунок 2 — Інтерфейс Wirecast

Ще одним важливим інструментом на ринку програмного забезпечення для багатоканальних трансляцій є OBS Studio (Open Broadcaster Software). OBS є безкоштовним рішенням з відкритим вихідним кодом, яке підтримує роботу з

кількома джерелами відео та аудіо, а також надає широкий спектр функцій для налаштування та персоналізації трансляцій. OBS підтримує інтеграцію з різними платформами стрімінгу, такими як YouTube, Twitch, Facebook, та надає можливість використовувати плагіни для розширення функціональних можливостей програми. Оскільки OBS є відкритою платформою, спільнота розробників постійно працює над створенням нових інструментів і доповнень, що дозволяють зробити програму ще більш функціональною. Проте, як безкоштовний інструмент, OBS може мати певні обмеження щодо обробки складних потоків у порівнянні з комерційними продуктами. Крім того, програма може потребувати певних технічних навичок для налаштування і оптимізації процесу трансляції, що може бути складним для новачків.

OBS Studio є одним із найбільш популярних безкоштовних програмних рішень для відеострімінгу, що надає користувачам потужні функції для захоплення та змішування відео та аудіо в реальному часі. Програма дозволяє створювати сцени, що складаються з різних джерел, таких як вікна програм, зображення, текст, браузери, вебкамери, захоплення відеокарт та багато іншого. Це дає користувачам безмежні можливості для створення індивідуальних, динамічних і професійно виглядаючих трансляцій. Однією з основних переваг

OBS Studio є його здатність до безперервного перемикання між кількома сценами, що можуть бути налаштовані з використанням індивідуальних анімаційних переходів. Це дозволяє оперативно адаптувати трансляцію під зміни контенту чи потреби аудиторії.

Програма також має інтуїтивно зрозумілий аудіо мікшер, який дозволяє застосовувати фільтри до кожного джерела звуку окремо. Серед доступних фільтрів — шумозаглушення, гейт для шумів та регулювання гучності, що допомагає підтримувати високу якість звуку без небажаних перешкод. Важливою особливістю є підтримка VST плагінів, що дає користувачам додаткові можливості

для обробки аудіо, дозволяючи налаштувати звук до найменших деталей. Цей рівень контролю є важливим для професійних трансляцій, де якість звуку має таке ж значення, як і відео.

Ще однією особливістю OBS Studio є потужні і зручні опції налаштування. Користувач може додавати нові джерела, дублювати існуючі або змінювати їх властивості з легкістю, що робить процес налаштування трансляцій максимально гнучким та ефективним. Для більш детального налаштування існує спрощена панель налаштувань, яка надає доступ до широкого спектра параметрів для налаштування кожного аспекту трансляції чи запису, що дає можливість точно адаптувати програму під конкретні вимоги користувача.

Модульний інтерфейс OBS Studio дає можливість користувачам налаштувати вікна інтерфейсу за своїми уподобаннями. Завдяки можливості переміщувати окремі панелі та вікна, користувач може створити робоче середовище, яке найкраще підходить для їхніх потреб. Кожну панель можна відкрити в окремому вікні, що дозволяє зручно розподілити елементи інтерфейсу на декількох моніторах або змінити розташування елементів для більш комфортної роботи.

OBS Studio підтримує всі популярні платформи для стрімінгу, включаючи YouTube, Twitch, Facebook і багато інших, що робить його універсальним інструментом для багатьох користувачів. Окрім цього, програма підтримує налаштування для захоплення та змішування відео в реальному часі з різних джерел, що дозволяє здійснювати трансляції високої якості на широкий спектр платформ.

Проте, незважаючи на величезні можливості, OBS Studio може бути менш інтуїтивно зрозумілим для новачків у порівнянні з іншими програмами, такими як Wirecast або vMix, оскільки має досить великий набір налаштувань, що потребує певного часу для освоєння. Крім того, OBS Studio не має вбудованих функцій,

таких як підтримка PTZ-камер чи вбудовані медіабібліотеки з готовими графічними елементами, що може обмежувати його функціональність для деяких специфічних задач, наприклад, для організації великих професійних подій.

У підсумку, OBS Studio є чудовим вибором для тих, хто шукає безкоштовне і потужне програмне забезпечення для відео- та аудіо-стрімінгу. Його гнучкість, розширені налаштування та підтримка безлічі платформ роблять його привабливим для користувачів, які хочуть максимально контролювати свої трансляції. Однак для великих, професійних подій, де потрібні додаткові функції, такі як підтримка PTZ або готові медіа-ресурси, варто розглянути інші варіанти програмного забезпечення.

Інтерфейс OBS Studio зображено на рисунку 3.

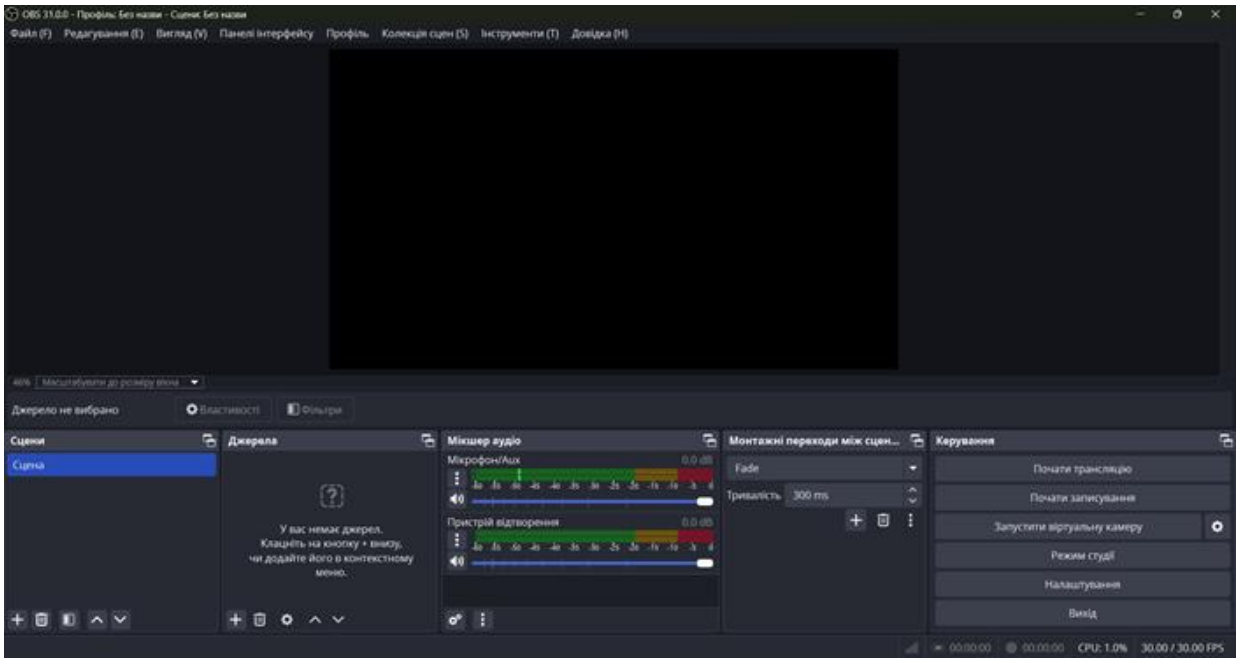

Рисунок 3 – Інтерфейс OBS Studio

Всі три програми — vMix, Wirecast та OBS Studio — пропонують потужні функції для проведення багатоканальних трансляцій, однак кожна з них має свої унікальні особливості, що можуть впливати на вибір залежно від вимог

користувача. vMix вирізняється своєю здатністю обробляти відео та аудіо високої якості, підтримкою складних сценаріїв в реальному часі, таких як накладення графіки, спецефектів та переходів. Крім того, він підтримує інтеграцію з Zoom, що є важливою функцією для трансляцій з великою кількістю учасників, а також забезпечує високу якість відео до 8K. Проте vMix вимагає потужного апаратного забезпечення та має високий поріг входу для новачків, що може бути недоліком для деяких користувачів.

Wirecast, у свою чергу, є більш гнучким і доступним для широкого кола користувачів завдяки вбудованим функціям, таким як підтримка хмарного стрімінгу, можливість створення анімованих графічних елементів та інтеграція з соціальними платформами. Wirecast також має вбудовану підтримку для віддалених гостей, що може бути важливим для проведення інтерв'ю або онлайн-конференцій. Однак його основним мінусом є те, що воно є платним, що може бути значним бар'єром для деяких користувачів, порівняно з OBS Studio, яке є безкоштовним.

OBS Studio виграє на фоні двох попередніх програм завдяки своїй доступності, при наявності потужних функцій для захоплення відео і аудіо. Він забезпечує широкі можливості для налаштування сцен, джерел і фільтрів, дозволяючи створювати високоякісні трансляції на платформі за допомогою модульного інтерфейсу. Однак OBS Studio може бути менш зручним для новачків, оскільки вимагає більше часу на освоєння, а також має деякі обмеження в порівнянні з vMix та Wirecast, зокрема відсутність підтримки деяких професійних функцій, таких як PTZ-камери чи вбудовані медіабібліотеки.

Загалом, вибір між цими трьома програмами залежить від специфічних потреб користувача. Якщо необхідна висока якість відео та професійні можливості для складних трансляцій, vMix стане найкращим вибором. Для більш доступного рішення з вбудованими функціями для стрімінгу та інтеграції з платформами

соціальних медіа, Wirecast є чудовим варіантом. OBS Studio ж залишається оптимальним рішенням для тих, хто шукає безкоштовне і гнучке програмне забезпечення для базових і середніх трансляцій, не потребуючи додаткових функцій високого рівня.

2 ПОСТАНОВКА ЗАДАЧІ, ОПИС ІНСТРУМЕНТІВ РОЗРОБКИ ТА ВИЗНАЧЕННЯ ПРИНЦИПІВ РОБОТИ ПРОГРАМИ

2.1 Етапи розробки програмного забезпечення

Розробка програмного забезпечення є складним і багатоступеневим процесом [13], що включає в себе різні етапи, кожен з яких має свою специфіку та значення. На рисунку 4 зображено основні етапи розробки програмного забезпечення, які допомагають структурувати процес і забезпечують ефективне досягнення цілей проекту.

Рисунок 4 — Основні етапи розробки програмного забезпечення

Зазначені етапи здебільшого виконуються поетапно, але на практиці можливі їхні перекриття та ітераційне виконання залежно від вибраної методології.

Збір вимог та аналіз [14] — це перший етап, що є основою для подальшої розробки ПЗ. На цьому етапі важливо зрозуміти потреби замовника, а також визначити обсяг та специфікації майбутнього програмного продукту. Розробники

працюють безпосередньо з клієнтом для збору детальної інформації про систему, що планується, і її функціональні вимоги. У цьому процесі використовуються різні методи комунікації, такі як інтерв'ю, опитування, аналіз існуючих систем або навіть спостереження за роботою користувачів. Одним із завдань цього етапу є визначення технічної доцільності проекту, а також оцінка можливості його реалізації в межах заданих часових та фінансових ресурсів.

Після того, як вимоги будуть зібрані та детально проаналізовані, наступним етапом є планування і проєктування. Тут створюється детальний план розробки, який включає в себе терміни, розподіл ресурсів і визначення основних етапів виконання проекту. Важливою складовою цього етапу є розробка архітектури програмного забезпечення, яка визначає основні компоненти системи та їх взаємодію. Також на цьому етапі розробляються елементи дизайну, зокрема інтерфейс користувача (UI) та досвід взаємодії з ним (UX). Хороший дизайн має забезпечити не лише естетичний вигляд, але й зручність та інтуїтивність користування продуктом, що значно підвищує його ефективність та задоволення від використання.

Коли проектування завершено і є чітке розуміння, як має виглядати кінцевий продукт, починається етап розробки, на якому здійснюється безпосереднє програмування. Програмісти пишуть код, створюють алгоритми, інтегрують різні модулі системи. На цьому етапі також активно використовуються методології гнучкої розробки, такі як Scrum або Kanban, що сприяють гнучкості і адаптивності розробки. У межах цих методологій робота над проектом здійснюється в коротших ітераціях (спринтах), що дозволяє швидко реагувати на зміни в вимогах і ефективно вирішувати проблеми, що виникають у процесі роботи.

Наступним важливим етапом є тестування та забезпечення якості, коли створений продукт піддається різним видам перевірок. Тестування дозволяє виявити та виправити помилки, а також перевірити, чи відповідає система

заявленим вимогам. На цьому етапі використовуються різноманітні види тестів, такі як модульне тестування, інтеграційне тестування, системне тестування та тестування прийнятності користувачем. Під час тестування [15] особлива увага приділяється перевірці на безпеку, ефективність та стабільність програмного забезпечення. Тестування дозволяє гарантувати, що продукт працює надійно та не містить серйозних дефектів, які можуть вплинути на його функціонування в реальних умовах.

Коли ПЗ пройшло тестування і всі критичні помилки усунені, наступним етапом є впровадження та реалізація продукту [16]. На цьому етапі програмне забезпечення переноситься в експлуатаційну середу, проводиться налаштування системи, а також здійснюється міграція даних, якщо це необхідно. Важливою частиною цього етапу є навчання кінцевих користувачів та надання відповідної документації. Для успішного впровадження продукту необхідно забезпечити правильне розгортання і налаштування програмного забезпечення, а також підготовити користувачів до роботи з новим продуктом, щоб забезпечити безперешкодну адаптацію.

Після того як програмне забезпечення стало доступним для кінцевих користувачів, наступним етапом є обслуговування та підтримка. Включає в себе усунення помилок, що виникають у процесі експлуатації, оновлення системи, виправлення уразливостей безпеки та реалізацію нових функцій, якщо це необхідно [17]. Підтримка також передбачає надання консультацій користувачам, а також постійне вдосконалення програмного продукту відповідно до нових вимог та змін у зовнішньому середовищі.

Методології розробки ПЗ визначають, як і коли кожен з цих етапів буде виконаний. Серед найбільш відомих методів розробки ПЗ можна виділити водоспадну модель (Waterfall), яка є лінійною та послідовною, а також методи, які включають ітераційні цикли, такі як Spiral, Scrum, Kanban та інші. Водоспадна

модель орієнтована на строгий порядок виконання етапів, де кожен наступний етап починається лише після завершення попереднього. Ідея полягає в тому, щоб всі етапи виконувалися без переглядів і змін, що дає точну структуру і передбачуваність, але має обмеження на гнучкість.

Водночас гнучкі підходи, такі як Scrum або Spiral, передбачають виконання роботи в циклах з регулярними переглядами та коригуваннями. Це дозволяє адаптувати процес під зміни вимог або середовища, що є особливо важливим у швидко змінюваних умовах розробки програмного забезпечення. Гнучкі методи також сприяють кращій взаємодії між розробниками та замовниками, що дозволяє краще зрозуміти потреби користувачів і забезпечити точніше виконання завдань.

2.2 Обгрунтування вибору середовища та мови програмування

Вибір середовища програмування є важливим етапом розробки програмного забезпечення, оскільки він безпосередньо впливає на продуктивність, зручність роботи розробників, а також на кінцеву якість продукту. Для багатьох розробників, особливо тих, хто працює з мовами програмування, орієнтованими на Microsoft, вибір середовища програмування часто падає на Visual Studio. Це середовище є одним з найпопулярніших і найбільш потужних інструментів для розробки програмного забезпечення в екосистемі Microsoft.

Visual Studio [18] — це інтегроване середовище розробки (IDE), яке забезпечує розробникам все необхідне для створення програмного забезпечення для різних платформ і пристроїв. Це включає в себе розробку для Windows, мобільних платформ на базі Android та iOS, а також для веб-додатків і серверних рішень. Visual Studio підтримує багато мов програмування, зокрема C#, VB.NET, C++, Python, F#, JavaScript, TypeScript та інші, що дозволяє використовувати одне середовище для різноманітних задач. Це робить його універсальним інструментом,

з яким розробники можуть працювати на різних проектах, без необхідності змінювати середовище для кожної нової технології.

Однією з ключових переваг Visual Studio є інтеграція з .NET Framework і .NET Core. Це дозволяє розробникам створювати високопродуктивні додатки для Windows, що є основною операційною системою для бізнес-інтерфейсів і настільних додатків. Для тих, хто працює з хмарними сервісами та розробляє додатки для Azure, Visual Studio надає зручні інструменти для інтеграції, що робить його ще більш привабливим для широкого кола розробників.

Однією з основних причин вибору Visual Studio є наявність великої кількості вбудованих інструментів, які полегшують процес розробки, тестування, налагодження та документування програмного забезпечення. Visual Studio має потужний редактор коду з підтримкою синтаксичного підсвічування, автозавершення, інтеграції з Git, а також відображенням помилок та попереджень під час написання коду. Це дозволяє знижувати кількість помилок, покращує продуктивність і забезпечує швидке написання чистого, зрозумілого коду. Потужний дебагер у Visual Studio дає змогу розробникам відстежувати виконання коду, зупиняти програми на потрібних точках і аналізувати зміни в змінних і стані програми в реальному часі. Це значно полегшує процес пошуку і виправлення помилок, що в свою чергу підвищує якість продукту. Visual Studio також надає багатий набір інструментів для юніт-тестування та інтеграційного тестування, таких як MSTest, NUnit, xUnit. Це дозволяє автоматизувати процес перевірки коректності програмного забезпечення, забезпечуючи його якість і стабільність на кожному етапі розробки. Крім того, Visual Studio є одним з основних середовищ для розробки додатків на платформі .NET, що робить його ідеальним вибором для проектів, які передбачають використання цієї технології. Воно забезпечує інтеграцію з усіма інструментами .NET, такими як Entity Framework, ASP.NET,

Xamarin для мобільних додатків та інші, що дозволяє розробникам швидко інтегрувати різні компоненти системи без необхідності шукати сторонні рішення.

Visual Studio підтримує масштабованість проектів, що дозволяє працювати як над малими, так і великими програмними продуктами. Завдяки своїм потужним інструментам для керування проектами, налагодження складних залежностей та інтеграції з іншими сервісами, розробка навіть великих корпоративних додатків стає набагато ефективнішою. Вбудовані інструменти для керування версіями, зокрема Git, дозволяють організувати командну роботу на проекті, що є важливою перевагою для великих колективів розробників.

Однією з основних переваг Visual Studio є підтримка багатьох платформ. Розробка додатків для Windows є основною функцією цього середовища, але також є можливість створювати мобільні додатки для платформ Android та iOS за допомогою Xamarin. Інтеграція з іншими середовищами і платформами, такими як Docker для контейнеризації, дозволяє ефективно створювати серверні додатки, а також забезпечує підтримку для розробки з використанням Kubernetes, що дозволяє створювати додатки, орієнтовані на хмарні рішення.

Для веб-розробки Visual Studio підтримує ASP.NET Core — потужну і швидку платформу для створення веб-додатків, а також Blazor для створення інтерактивних веб-додатків на C#. Це дає можливість створювати не лише традиційні клієнт-серверні додатки, але й додатки, які безпосередньо виконуються в браузері, завдяки технології WebAssembly.

Що стосується мови програмування, то найбільш оптимальним вибором в даному випадку є C#. Ця мова є основною для розробки додатків на платформі .NET і має безліч переваг, які роблять її ідеальним вибором для створення високопродуктивних і надійних програмних рішень. Однією з головних причин, чому C# є настільки популярним, є його багатофункціональність і зручність для розробників. Мова поєднує в собі високу продуктивність, простоту синтаксису і

потужні можливості для створення різноманітних додатків, від простих десктопних до складних веб- і мобільних програм.

C# надає широкий набір функцій для розробки безпечного і ефективного коду. Підтримка об'єктно-орієнтованого програмування (ООП) дозволяє розробникам створювати масштабовані і легко підтримувані програми. Крім того, C# підтримує функціональні можливості, такі як LINQ (Language Integrated Query), що значно спрощує роботу з даними та запитами, а також асинхронне програмування, яке дозволяє розробляти швидкодіючі і ефективні програми, що працюють з великими обсягами даних або в умовах обмежених ресурсів.

Іншою важливою перевагою C# є його тісна інтеграція з платформою .NET, [19] що забезпечує доступ до великої кількості бібліотек і фреймворків. Це дозволяє розробникам використовувати готові рішення для рутинних задач, таких як робота з базами даних, мережеві запити, безпека, а також створення графічних інтерфейсів. Наприклад, Entity Framework дозволяє легко працювати з базами даних, а ASP.NET надає потужні інструменти для розробки веб-додатків. Xamarin дає можливість створювати кросплатформенні мобільні додатки, що працюють на Android і iOS, використовуючи один кодовий базис.

Ще однією важливою характеристикою C# [20] є його підтримка багатозадачності і паралельного програмування, що є критично важливим для створення сучасних високопродуктивних додатків. За допомогою таких конструкцій, як async/await, розробники можуть створювати ефективні асинхронні додатки, які легко справляються з операціями вводу/виводу, не блокуючи основний потік виконання програми. Крім того, багатопоточність і асинхронне програмування дозволяють максимально використовувати можливості багатоядерних процесорів, що значно підвищує продуктивність програм.

Завдяки своїй простоті, потужності та широким можливостям, C# є ідеальним вибором для розробки додатків на платформі .NET. Він дозволяє

розробникам створювати високоякісні, ефективні і безпечні програми, використовуючи один з найпопулярніших і найбільш підтримуваних технологічних стеків в світі. Крім того, C# має активну і велике співтовариство розробників, що забезпечує постійну підтримку та доступ до величезної кількості ресурсів, включаючи документацію, приклади коду, бібліотеки та форуми для обміну досвідом. Усе це робить C# найкращим вибором для сучасної розробки програмного забезпечення, зокрема для створення додатків, які працюють на платформі .NET.

Що стосується вибору технології для розробки графічних інтерфейсів, то для проектів, орієнтованих на платформу .NET, найкращим варіантом є WPF (Windows Presentation Foundation) [21]. WPF є потужною технологією для створення настільних додатків на платформі Windows, яка забезпечує велику гнучкість у створенні графічних інтерфейсів користувача. Вона дозволяє розробникам створювати як прості, так і складні інтерфейси з використанням багатих графічних елементів, анімацій, візуальних ефектів і інтерактивних елементів, що значно підвищує зручність роботи кінцевого користувача.

Однією з головних переваг WPF є її можливості для створення складних і красивих інтерфейсів з використанням XAML (Extensible Application Markup Language) [22]. XAML є декларативною мовою розмітки, яка дозволяє відокремити логіку програми від її вигляду, що сприяє кращій організації коду і полегшує його підтримку. За допомогою XAML можна описувати зовнішній вигляд елементів інтерфейсу, таких як кнопки, текстові поля, панелі та інші компоненти, без необхідності писати великі обсяги коду.

WPF підтримує концепцію прив'язки даних (data binding) [23], що дозволяє легко зв'язувати елементи інтерфейсу з даними з бізнес-логіки. Це дозволяє значно зменшити кількість коду, який потрібно писати для обробки подій та взаємодії з користувачем. Прив'язка даних також підтримує двосторонній зв'язок, що означає,

що зміни в інтерфейсі користувача автоматично оновлюють відповідні значення в коді програми і навпаки. Такий підхід спрощує створення динамічних інтерфейсів, де дані змінюються в реальному часі.

Крім того, WPF забезпечує потужні можливості для роботи з графікою, що є важливою складовою сучасних інтерфейсів користувача. Завдяки використанню DirectX для рендерингу, WPF може створювати складні візуальні ефекти, анімації і графіку в реальному часі з високою продуктивністю. Це дозволяє розробникам створювати додатки з красивими анімаціями, ефектами переходів, прозорістю та іншими графічними елементами, які роблять користування програмою більш приємним і інтуїтивно зрозумілим.

Окрім того, WPF має вбудовану підтримку стилів та шаблонів, що дає можливість створювати унікальні і кастомізовані інтерфейси без необхідності повторювати код для кожного елемента. Стилі дозволяють задавати вигляд елементів інтерфейсу (наприклад, кольори, шрифти, розміри) в одному місці, що спрощує зміну вигляду програми і дозволяє швидко адаптувати її до різних вимог.

WPF також підтримує використання шаблонів для елементів управління, що дозволяє змінювати їх вигляд без впливу на функціональність. Це дає розробникам можливість створювати унікальні елементи інтерфейсу, які відповідають специфічним вимогам проекту, при цьому зберігаючи загальну архітектуру програми.

Ще однією перевагою WPF є його здатність до масштабування і адаптації до різних роздільних здатностей і розмірів екранів. За допомогою механізмів, таких як автопозиціонування елементів і прив'язка до розмірів вікна, можна створювати інтерфейси, які будуть коректно відображатися на різних пристроях і екранах з різною роздільною здатністю. Це дозволяє значно покращити досвід користувача при використанні програми на різних платформах, зокрема на комп'ютерах з різними розмірами екрану.

Враховуючи всі ці можливості, WPF є ідеальним вибором для створення додатків на платформі Windows з графічними інтерфейсами, оскільки він надає всі необхідні інструменти для розробки сучасних, красивих і високопродуктивних програм. Поєднання з C# і платформою .NET робить WPF потужним інструментом для створення надійних і масштабованих рішень, що задовольняють потреби користувачів і вимоги бізнесу.

2.3 Постановка задачі

Необхідно розробити інтегровану комп'ютерну систему для проведення інтерактивних багатоканальних трансляцій, яка забезпечить ефективне управління різними медіа-потоками в реальному часі. Система повинна бути реалізована у вигляді програми для робочих станцій, що дозволяє користувачам організовувати і керувати трансляціями, інтегруючи різні потоки даних, такі як відео, аудіо, текст та зображення. Програма повинна надавати можливість роботи з кількома потоками одночасно, з підключенням до різних пристроїв, таких як камери, мікрофони та суфлери, що дозволяє відображати ці дані на інтерактивних слайдах у реальному часі.

Користувачі програми повинні мати можливість додавати нові медіа-потоки на слайди, налаштовувати їх розташування і змінювати параметри відображення, забезпечуючи динамічний контент для трансляції. Крім того, програма повинна мати функціонал для налаштування параметрів трансляцій, таких як вибір джерел даних, синхронізація потоків та їх подальша передача на інші пристрої для публічного перегляду.

Програма складається з таких основних компонентів:

— основна панель, яка відображає контент;

— панель, яка відображає усі наявні слайди та дає можливість їх створювати;

— панель, яка відображає усі наявні потоки і дає можливість їх створювати;

— діалогове вікно для налаштування потоків.

Діаграму станів зображено на рисунку 5.

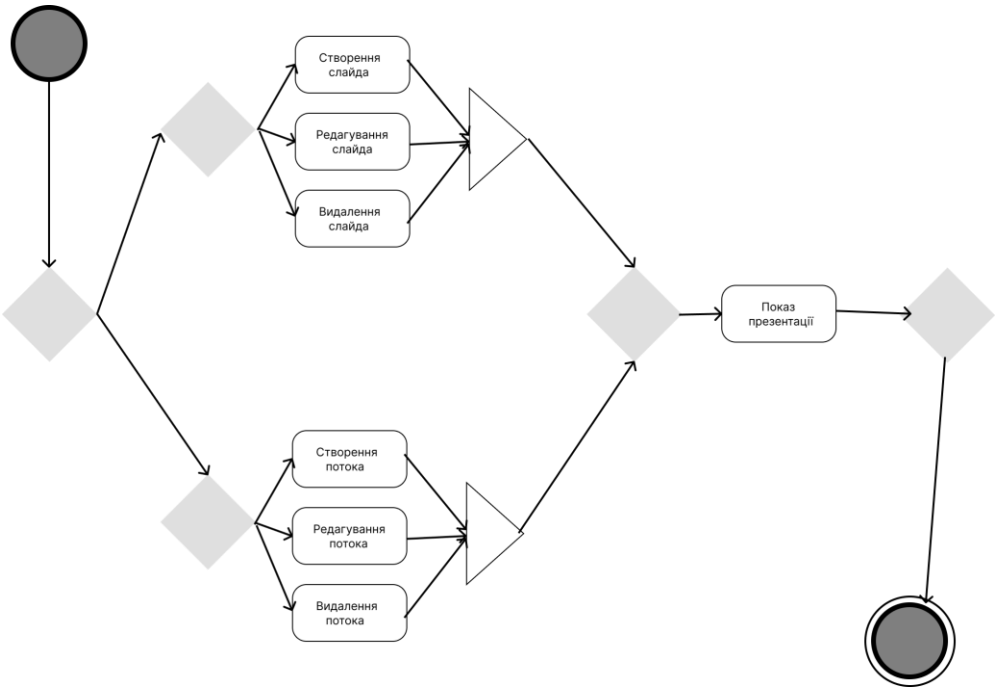

Рисунок 5 — Діаграма станів проекту

3 РЕАЛІЗАЦІЯ ПРОГРАМИ

3.1 Створення дизайну

У процесі розробки дизайну програми "MultiStream" передбачено створення інтуїтивно зрозумілого та функціонального інтерфейсу, який забезпечить користувачам зручний доступ до основних можливостей системи. Дизайн планується організувати таким чином, щоб користувач міг легко керувати слайдами, потоками даних і редагувати їх у реальному часі, зберігаючи простоту та естетичність робочого простору [24].

Інтерфейс програми буде поділено на дві основні зони. Ліва частина міститиме панелі для управління слайдами та потоками. У верхній частині планується розташувати список доступних слайдів, який дозволятиме переглядати їх у вигляді текстового списку. Кожен елемент списку відображатиме коротку інформацію про відповідний слайд. Для зручності користувачів буде додано можливість сортування та швидкого доступу до редагування. Під списком слайдів розміщуватиметься кнопка для створення нових слайдів. Нижня частина лівої панелі буде присвячена роботі з потоками даних. Передбачено список потоків із подібними функціями, а також кнопку для додавання нових потоків.

Центральна частина інтерфейсу стане робочою зоною для редагування та перегляду слайдів. У центрі планується розташувати велику область відображення, що за замовчуванням буде прихована до моменту вибору слайда для редагування. Вибраний слайд буде відображатися у вигляді полотна на контрастному тлі, що забезпечить комфорт під час роботи. Це полотно виконуватиме роль інтерактивної області, де користувачі зможуть додавати, змінювати та переміщувати об’єкти.

Для створення приємного візуального враження вибрано темну колірну гаму, що складається з глибокого сірого та чорного відтінків для основного тла та панелей. Вона дозволяє зменшити навантаження на очі під час тривалої роботи. Контрастні елементи, такі як кнопки й текст, будуть виконані у світлих тонах, щоб

забезпечити чіткість і зручність сприйняття. Крім того, передбачено чітке розділення зон за допомогою обрамлення та відступів, щоб уникнути візуального перевантаження.

Дизайн буде реалізовано із врахуванням принципів адаптивності, що дозволить програмі коректно відображатися на екранах із різними розмірами. Планується використання WPF як основного інструменту для створення інтерфейсу, що забезпечить високу гнучкість і можливість масштабування.

3.2 Реалізація дизайну засобами XAML

Після створення проекту, перше, що потрібно зробити – це налаштувати головне вікно. За замовчуванням, у файлі MainWindow.xaml уже створено тег <Window> з низкою стандартних атрибутів.

Атрибут x:Class="MultiStream.MainWindow" вказує на пов’язаний клас коду, який відповідає за логіку вікна. У цьому випадку це клас MainWindow, розташований у просторі імен MultiStream. Він автоматично створюється в файлі MainWindow.xaml.cs і використовується для роботи з логікою взаємодії користувача з інтерфейсом.

Атрибут xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation" визначає стандартний простір імен для елементів XAML, які належать до бібліотеки WPF. Це обов’язковий атрибут, оскільки всі елементи інтерфейсу WPF базуються на цьому просторі імен.

Атрибут xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" використовується для роботи зі специфічними функціями XAML, такими як посилання на ресурси, оголошення імен або використання прив’язок.

Атрибути xmlns:d та xmlns:mc призначені для підтримки інструментів розробки, таких як дизайнер XAML. Атрибут xmlns:d="http://schemas.microsoft.com/expression/blend/2008" використовується для

додавання елементів, які потрібні лише під час дизайну, але ігноруються під час компіляції. Атрибут xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" разом із властивістю mc:Ignorable="d" дозволяє вказати, що простір імен d буде проігнорований під час виконання програми. Це забезпечує зручність роботи в середовищах розробки, таких як Visual Studio [25].

У наступних кроках налаштовуються параметри вигляду вікна. Наприклад, властивість Title="MultiStream" визначає текст заголовка вікна, який відображатиметься у верхній частині. Параметр WindowState="Maximized" задає стартовий стан вікна – воно відкривається в розгорнутому вигляді, що забезпечує максимальне використання доступного простору. Атрибут WindowStyle="SingleBorderWindow" встановлює стандартний стиль вікна з рамкою та системною панеллю, що включає кнопки закриття, згортання та розгортання.

Значення властивості Background="#1E1E1E" задає фон вікна, використовуючи темний колір із шістнадцятковим кодом. Цей колір є базовим для створення контрастного та сучасного інтерфейсу.

Ці базові налаштування є відправною точкою для подальшого розширення та адаптації інтерфейсу під функціональні вимоги проекту.

XAML надає розробникам багатий набір контейнерів для компонування інтерфейсу користувача, що дозволяє гнучко організовувати елементи у вікні програми. Контейнери в XAML [26] відповідають за розташування, розмір і взаємодію графічних компонентів на формі. Основними типами контейнерів є Grid, StackPanel, WrapPanel, DockPanel та Canvas, кожен із яких має свої особливості та сценарії використання.

Grid є одним із найпотужніших і найпоширеніших контейнерів для компонування інтерфейсу. Він дозволяє створювати таблицю з рядків і стовпців, де можна чітко вказати розмір кожного елемента через властивості RowDefinitions

та ColumnDefinitions. Grid є оптимальним вибором для складного компонування, де важлива точність і фіксовані межі між елементами.

StackPanel використовується для послідовного розміщення елементів у вертикальному або горизонтальному напрямку. Елементи автоматично додаються один за одним у порядку їх оголошення, що робить цей контейнер зручним для простих і лінійних інтерфейсів. Для зміни напряму компонування достатньо задати властивість Orientation.

WrapPanel схожий на StackPanel, але з тією відмінністю, що коли елементи не поміщаються у доступний простір по горизонталі або вертикалі, вони автоматично переносяться на наступний рядок або стовпець. Цей контейнер зручний для динамічного відображення списків чи груп компонентів зі змінними розмірами.

DockPanel забезпечує можливість прив’язки елементів до певних сторін контейнера: ліворуч, праворуч, зверху чи знизу. Завдяки цьому елементи можна розташовувати відносно країв вікна або інших компонентів. Якщо один елемент не закріплено, він займає весь вільний простір, що залишився.

Canvas надає найвищий рівень контролю над позиціонуванням елементів, оскільки вони розташовуються за абсолютними координатами. Кожен об’єкт у Canvas може бути зміщений за допомогою властивостей Canvas.Left та Canvas.Top. Цей контейнер є ідеальним для створення інтерфейсів, де важливо точно визначити позицію елементів, наприклад у випадку інтерактивних середовищ чи анімацій.

Таким чином, вибір контейнера залежить від потреб конкретного інтерфейсу. Для лінійного розташування елементів найкраще підходить StackPanel, а для складного компонування з можливістю розбиття на рядки та стовпці — Grid. Якщо потрібна динамічна адаптація елементів, ефективним рішенням є WrapPanel, тоді як для фіксованого прив’язування до країв — DockPanel. У разі, коли необхідно повністю контролювати координати елементів, слід використовувати Canvas.

Після вибору необхідних контейнерів можна ефективно розташувати всі елементи інтерфейсу відповідно до вимог програми, що забезпечить зручність і логічність структури компонування.

В даному випадку, за основу було взято контейнер Grid, аби розділити вікно на кілька областей і забезпечити чітке компонування елементів інтерфейсу. Використання Grid дозволяє створити структуру з рядків та стовпців, що дає можливість точніше керувати розміщенням елементів. У нашому випадку було визначено два стовпці, де один займає меншу частину вікна для лівої панелі з елементами управління, а інший — більшу частину для основного вмісту, де відображається слайд або потік. Такий поділ дозволяє зручно організувати взаємодію між елементами і забезпечити зручність користування інтерфейсом.

Для кожного стовпця та ряду в Grid визначені відповідні розміри через властивості ColumnDefinitions та RowDefinitions. Це дозволяє чітко налаштувати, яку частину простору повинні займати конкретні елементи. Так, ліворуч розміщуються панелі для списків слайдів та потоків, які займають невелику частину вікна, а праворуч — великий контейнер для відображення вмісту слайда або потоку.

У лістингу 1 показано код, який визначає структуру контейнера Grid за допомогою ColumnDefinitions та RowDefinitions, які дозволяють розділити вікно на кілька областей. У Grid можна вказати кількість стовпців і рядків, а також визначити, скільки простору вони повинні займати в межах батьківського елемента. У даному випадку, перший стовпець має ширину, що займає 25% від доступної ширини вікна, а другий стовпець займає 75% від цієї ж ширини. Висота кожного рядка дорівнює 50% від доступної висоти. Отже, вікно розподіляється на дві вертикальні частини: ліву, яка займає 25% ширини, та праву, що займає 75%, і дві рівні частини по висоті, кожна з яких займає 50%. Це дозволяє організувати елементи у вікні так, щоб вони відповідали вказаному розподілу простору.

Лістинг 1 — Структура контейнера Grid

```
<Grid.ColumnDefinitions>
<ColumnDefinition Width="25*" />
<ColumnDefinition Width="75*" />
</Grid.ColumnDefinitions>
<Grid.RowDefinitions>
<RowDefinition Height="50*" />
<RowDefinition Height="50*" />
</Grid.RowDefinitions>
```

А на лістингу 2 показано код лівої верхньої частини програми, що містить список слайдів та кнопку для додавання нового слайду. Ця частина інтерфейсу розташована в межах StackPanel, який знаходиться в першій колонці і першому рядку сітки, наданої контейнером Grid. StackPanel дозволяє організувати елементи вертикально, що забезпечує компактне і зручне розташування елементів, зокрема списку та кнопки.

Лістинг 2 — Код лівої верхньої частини програми

```
<StackPanel Grid.Column="0" Grid.Row="0" Background="#2C2C2C"
Margin="5">
<ListBox x:Name="SlideList"
Foreground="White"
Background="#333333"
BorderThickness="1"
BorderBrush="#444444"
VerticalAlignment="Top"
Height="290"
ScrollViewer.VerticalScrollBarVisibility="Auto"
SelectionChanged="SlideList_SelectionChanged"
PreviewKeyDown="SlideList_PreviewKeyDown">
<!-- Шаблон елемента списку -->
<ListBox.ItemTemplate>
<DataTemplate>
<StackPanel>
<TextBlock x:Name="SlideName" Foreground="White" Padding="5" />
</StackPanel>
</DataTemplate>
```

```
</ListBox.ItemTemplate>
</ListBox>
<Button x:Name="AddSlideBTN"
Content="Додати слайд"
Foreground="White"
Background="#444444"
Margin="5"
Padding="10"
HorizontalAlignment="Stretch"
Click="AddSlideBTN_Click" />
</StackPanel>
```

Першим елементом у StackPanel є ListBox, який представляє список слайдів. Він має визначений колір фону #333333, що створює контраст з білим текстом, що робить інформацію більш помітною. Багатий набір властивостей налаштовує відображення списку: висота компонента встановлена на 290 пікселів, а вертикальна смуга прокручування буде з'являтися автоматично, якщо кількість елементів перевищує висоту, відведену для списку. Це робить інтерфейс адаптивним і зручним у використанні навіть при великій кількості слайдів.

Однією з важливих властивостей цього елемента є прив'язка подій SelectionChanged та PreviewKeyDown. Подія SelectionChanged дозволяє обробляти зміни в виборі елементів списку, що важливо для реагування на вибір конкретного слайду користувачем. Подія PreviewKeyDown дає можливість реагувати на натискання клавіш, наприклад, для здійснення навігації по списку або інших дій.

Шаблон елемента списку в ListBox використовується для відображення кожного слайду. Для цього у ItemTemplate задається StackPanel, в якому знаходиться TextBlock для відображення імені слайду. Всі елементи мають білий колір тексту, що контрастує з темним фоном списку, що дозволяє забезпечити легкість сприйняття інформації. Відступи на елементах забезпечують зручне розташування тексту, що підвищує читабельність.

Під ListBox розташовується кнопка AddSlideBTN, яка служить для додавання нового слайду до списку. Ця кнопка має стиль, схожий на стиль елементів списку, з білим текстом на темному фоні #444444, що підвищує візуальну цілісність інтерфейсу. Властивість Padding додає додатковий простір навколо тексту кнопки для забезпечення кращого візуального ефекту та полегшення натискання на мобільних пристроях або з іншими формами введення. Кнопка також налаштована так, щоб розтягуватися по ширині доступного простору завдяки властивості HorizontalAlignment="Stretch", що дає змогу досягти однакової ширини кнопки з іншими елементами інтерфейсу.

Подія Click кнопки прив'язана до методу AddSlideBTN_Click, що означає, що при натисканні на кнопку буде виконуватися код, пов'язаний з додаванням нового слайду до списку. Це створює функціональний механізм для користувача, який дозволяє швидко додавати нові елементи до програми через просту і зрозумілу кнопку.

Загалом, ця частина інтерфейсу є елементом управління списками, що дозволяє користувачеві ефективно працювати зі списком слайдів і додавати нові елементи. Вибір елементів списку та взаємодія з ними мають бути інтуїтивно зрозумілими завдяки продуманому дизайну, який дозволяє мінімізувати кількість дій користувача для виконання основних операцій.

Результат такої розмітки XAML показано на рисунку 6.

Після того, як за таким же принципом було створено список та кнопку для потоків, можна перейти до створення основної робочої області. На лістингу 3 показано код XAML.

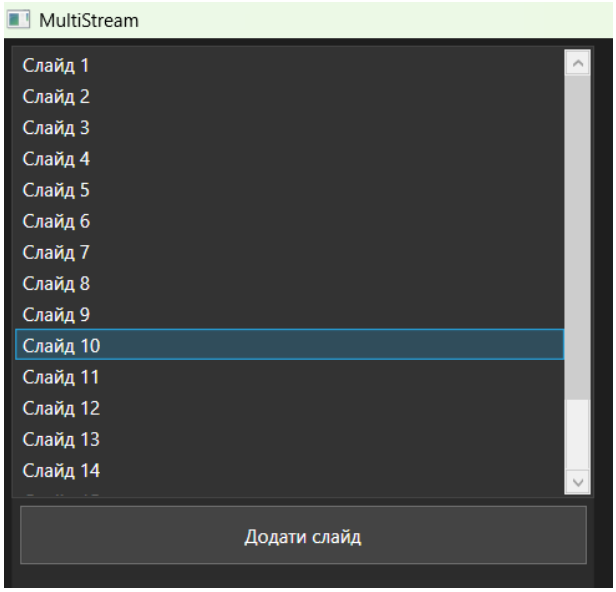

Рисунок 6 — Інтерфейс списку слайдів

Лістинг 3 - Код XAML для створення основної робочої області

```
<Grid Grid.Column="1" Grid.RowSpan="2" Background="#1E1E1E"
Margin="5">
<Border Background="#2C2C2C"
BorderThickness="1"
BorderBrush="#444444"
HorizontalAlignment="Center"
VerticalAlignment="Center"
Width="800"
Height="450">
<Canvas x:Name="slideCanvas"
Background="White"
Visibility="Collapsed" />
</Border>
</Grid>
```

Цей уривок відповідає за створення правої частини інтерфейсу програми, в якій відображатиметься сам слайд. Основним контейнером для цієї частини є Grid, що розташовується в другій колонці та займає обидва рядки сітки завдяки властивості Grid.RowSpan="2". Це означає, що елемент займе всю висоту вікна з

верхнього та нижнього рядка, що дає змогу створити більш просторий блок для відображення слайдів. Контейнер Grid має фон кольору #1E1E1E (темно-сірий) та відступи в 5 пікселів від межі.

В середині Grid розташований Border, який має більш яскравий фон #2C2C2C (сірий середнього тону) і темний бордер #444444. Це дозволяє створити візуальний контур для елемента, чітко виділяючи його на фоні всього інтерфейсу. Border має фіксовані розміри — ширина 800 пікселів і висота 450 пікселів, що відповідає типовим пропорціям для зображень або відео, з якими працюватиме програма.

У середині Border знаходиться Canvas — це основний контейнер для візуального відображення контенту слайда. Canvas використовує білий фон, що забезпечує чистий фон для зображень або відео, які будуть відображені в цьому полі. Важливою властивістю є Visibility="Collapsed", що означає, що на старті елемент не буде видимим. Це може бути корисно, коли вміст ще не завантажено або якщо слайд повинен з'являтися після певної дії користувача.

Canvas дає змогу динамічно додавати різні елементи в конкретні координати, що є важливим для відображення слайдів з різними типами контенту (текст, зображення, відео) на конкретних позиціях. Це також дозволяє реалізувати складніші анімації або анімовані ефекти при переміщенні слайдів. І загалом, інтерфейс виглядає так, як показано на рисунку 7.

3.3 Розробка функціоналу програми

У програмі, що реалізує функціонал для роботи з слайдами, користувач має можливість додавати, редагувати та видаляти слайди. Кожен слайд є окремим об’єктом, який відображається на екрані у вигляді елемента Canvas. Усі слайди зберігаються в колекції, що дозволяє здійснювати ефективне управління ними. Програма використовує елементи управління WPF, такі як ListBox, Canvas та Button, що забезпечують інтуїтивно зрозумілий інтерфейс для користувача.

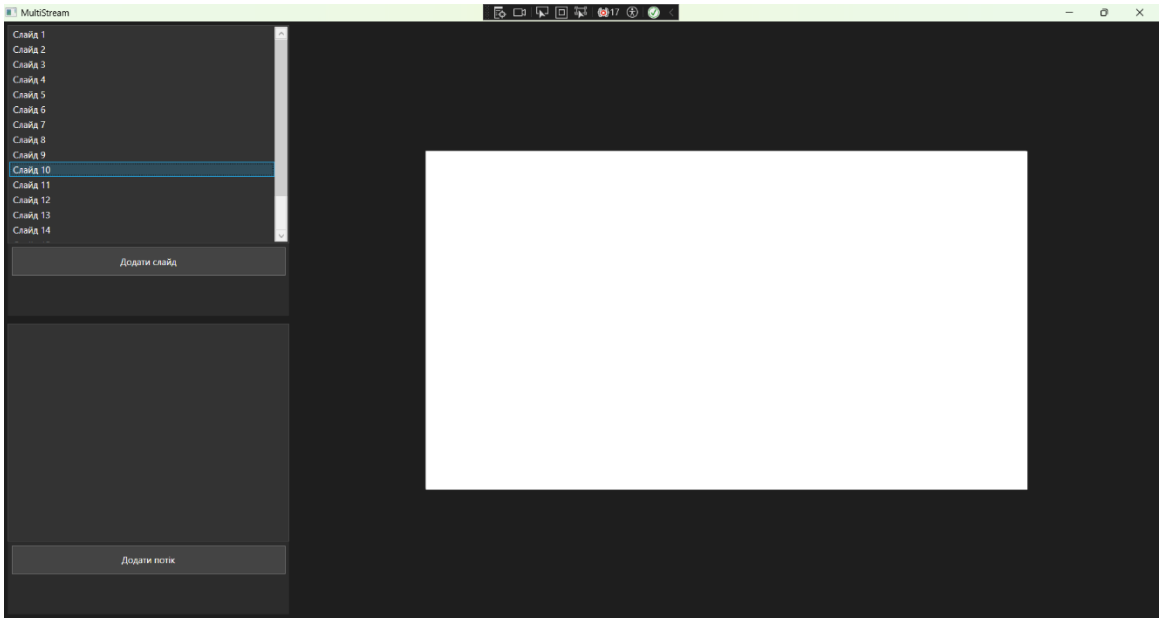

Рисунок 7 — Загальний вигляд інтерфейсу програми

Основна мета цієї частини програми — надати можливість створювати нові слайди, редагувати їхні назви та видаляти їх за потребою. Для реалізації цього функціоналу в програмі використовуються кілька ключових методів, що обробляють дії користувача.

Коли користувач натискає кнопку для додавання нового слайда, викликається метод AddSlideBTN_Click. Цей метод відповідає за кілька операцій. По-перше, кожного разу, коли додається новий слайд, збільшується лічильник слайдів, що зберігається в змінній _slideCounter. Це дозволяє генерувати унікальні імена для слайдів, у форматі "Слайд X", де X — це поточний номер слайда. Завдяки цьому кожен новий слайд отримує окреме ім’я, що легко ідентифікується в списку слайдів. Це є важливим аспектом для підтримки організації в програмі, оскільки кожен слайд має свій унікальний ідентифікатор, і програма не буде стикатися з конфліктами імен.

Далі програма створює новий об'єкт Canvas, який є контейнером для вмісту слайда. Встановлюються розміри цього канвасу (800 на 450 пікселів), що є стандартним розміром для слайдів, та фон, який буде білим, щоб користувач міг

зручно працювати з елементами на слайді. Новий Canvas додається до колекції _slides, де ім’я слайда виступає ключем, а сам об'єкт Canvas — значенням. Це дозволяє зберігати всі слайди в пам’яті та забезпечувати швидкий доступ до них за допомогою їхніх імен.

Після створення нового слайда програма додає елемент до списку слайдів, що відображається в інтерфейсі користувача через ListBox. Кожен елемент списку є об'єктом ListBoxItem, який містить текстову інформацію про слайд. Для кожного нового слайда створюється новий елемент ListBoxItem, який представляє собою назву слайда. Цей елемент додається до SlideList, що є списком слайдів, і таким чином користувач бачить новий слайд у списку. Для кожного елемента списку визначена подія, яка спрацьовує при двократному натисканні на елемент. Ця подія викликає метод EditSlideName, який дозволяє редагувати назву слайда, змінюючи її безпосередньо в інтерфейсі.

Метод SlideList_SelectionChanged обробляє ситуацію, коли користувач вибирає слайд зі списку. Коли слайд вибрано, програма перевіряє, чи є відповідний об'єкт Canvas у колекції слайдів. Якщо слайд знайдений, він відображається на екрані, очищаючи попередній вміст, і робиться видимим. Якщо слайд не вибрано або якщо всі слайди видалено, то програма приховує вміст Canvas. Таким чином, реалізується можливість перемикання між слайдами та їхнім відображенням.

Для редагування назви слайда використовується метод EditSlideName, який запускається при двократному натисканні на елемент списку. У цьому методі створюється нове текстове поле (TextBox), яке дозволяє користувачу ввести нову назву для слайда. Текстове поле автоматично отримує фокус і дозволяє користувачеві змінити назву слайда без зайвих кроків. Після того, як користувач змінює назву, програма перевіряє, чи є вже слайд з такою назвою. Якщо такої назви немає, програма оновлює колекцію слайдів, замінюючи старе ім’я на нове. Якщо

нова назва є порожньою або вже використовується для іншого слайда, програма зберігає попередню назву.

Іншою важливою функцією є можливість видалення слайдів. Коли користувач натискає клавішу Delete на клавіатурі, викликається метод SlideList_PreviewKeyDown. Цей метод перевіряє, чи є вибраний елемент у списку слайдів. Якщо елемент вибрано, програма видаляє відповідний слайд з колекції _slides. Після цього програма перевіряє, чи був слайд відображений на екрані, і якщо так, очищає його вміст і приховує. Крім того, якщо всі слайди видалено, програма приховує полотно слайда (Canvas). Цей механізм видалення дозволяє користувачу ефективно працювати з великими списками слайдів, не переживаючи про їхнє повторне додавання чи збереження.

Програма має інтуїтивно зрозумілий інтерфейс, де всі слайди зберігаються в одному місці, і користувач може легко додавати нові слайди, змінювати їхні назви та видаляти їх без складних процедур. Програма також передбачає механізми перевірки на наявність помилок, що дозволяє уникнути непередбачуваних ситуацій. Усі ці функції забезпечують зручну роботу з слайдами та дозволяють користувачу зосередитися на створенні контенту, не витрачаючи часу на складні налаштування або операції з редагуванням.

Загалом, основними особливостями програми є її здатність до динамічного додавання та редагування слайдів, підтримка інтуїтивного інтерфейсу для користувача та здатність легко управляти списком слайдів. Ці функції забезпечують ефективність і зручність використання програми, що робить її корисним інструментом для створення та демонстрації мультимедійних презентацій.

Функція AddStreamBTN_Click є ключовим елементом інтерфейсу користувача для додавання потоків у програму. Ця функція викликається при натисканні кнопки на інтерфейсі програми і здійснює додавання нових елементів

(потоків) до списку вікон або слайдів, які відображають різні типи медіа. Вона є частиною більш широкої системи, що дозволяє користувачеві інтерактивно додавати мультимедійні елементи, такі як зображення, відео та відеопотоки з камери, на слайди у вікні програми. Функція працює за асинхронним принципом, де взаємодія з інтерфейсом користувача та завантаження медіа елементів відбувається в окремих потоках, що забезпечує плавну роботу програми. Розглянемо детально всі етапи її виконання.

На початку функція створює екземпляр вікна AddStreamWindow, яке є окремим діалоговим вікном для введення параметрів нового потоку. Цей об'єкт використовується для того, щоб користувач міг вказати необхідні параметри для потоку, такі як назва потоку та джерело медіа. Виклик методу ShowDialog() відкриває це вікно в модальному режимі, що означає, що поки користувач не зробить вибір або не закриє вікно, програма не може продовжити свою роботу. Після того як користувач вибирає джерело та вводить назву потоку, результат цих дій передається назад до головного вікна.

Коли користувач закриває вікно, програма перевіряє, чи дійсно була підтверджена операція (тобто чи не було натиснуто кнопку "Скасувати"). Якщо було обрано джерело медіа і підтверджено вибір, то програма переходить до наступного етапу — додавання нового потоку в список потоків. Для цього створюється новий елемент ListBoxItem, який представляє собою один потік у списку потоків на інтерфейсі. Контент цього елемента формується як комбінація назви потоку та типу джерела, яке вказав користувач. Наприклад, якщо користувач вибрав зображення, яке знаходиться за шляхом "C:\images\photo.jpg", то вміст елемента буде виглядати так: "Фото (C:\images\photo.jpg)". Колір тексту елемента встановлюється білим, щоб текст був добре видимий на темному фоні.

Далі, в залежності від того, який тип джерела вибрав користувач, виконується відповідна обробка цього потоку. Якщо це зображення, то програма

виконує перевірку формату файлу, перевіряючи розширення джерела. Якщо розширення файлу відповідає одному з типів зображень (наприклад, .jpg, .png, або .bmp), програма намагається завантажити це зображення у форматі BitmapImage. Це здійснюється за допомогою конструктора цього класу, в якому передається URI до файлу. Це завантаження виконується асинхронно, але в даному випадку програма чекає, поки зображення буде готове до відображення, перш ніж продовжити виконання.

Після того як зображення завантажено, воно вставляється на відповідний слайд у вікні програми. Для цього створюється об'єкт Image, який є елементом керування в WPF, що дозволяє відображати зображення. Цей елемент додається на конкретний слайд (наприклад, перший слайд) в колекцію дітей елемента Canvas. Слайд — це контейнер, в який можна додавати різні елементи, включаючи зображення, текст, відео тощо. Всі попередні елементи на слайді очищуються перед додаванням нового зображення. Це забезпечує, що на слайді буде відображено тільки одне зображення в кожен момент часу.

Якщо вибраним джерелом є відео (формати .mp4, .avi), то програма обробляє цей потік відповідно. Вона створює об'єкт MediaElement, який дозволяє відтворювати відео на слайді. Це елемент WPF, що використовується для відображення медіа-контенту, зокрема відео. Відео завантажується через URI, вказаний користувачем. У цьому випадку відео не тільки додається на слайд, але й автоматично запускається для відтворення. У властивості LoadedBehavior встановлюється значення Manual, що означає, що відео почне відтворюватися лише після того, як ми викличемо метод Play(). Таким чином, відео на слайді починає відтворюватися лише після того, як воно буде повністю завантажено і готове до відтворення.

Якщо користувач вибирає камеру як джерело потоку, програма повинна ініціалізувати відеозахоплення з камери. Для цього використовується об'єкт

MediaCapture, який є частиною бібліотеки UWP і дозволяє працювати з медіа-пристроями, такими як камери та мікрофони. Спочатку ініціалізується об'єкт MediaCapture, після чого асинхронно налаштовується його використання для захоплення відео. Якщо ініціалізація проходить успішно, створюється об'єкт CaptureElement, який відповідає за відображення відео з камери на екрані. Цей елемент додається до слайда, і відео з камери починає відображатися. Важливо, що відео з камери виводиться в реальному часі, тому програма повинна бути налаштована таким чином, щоб відео показувалося без затримок.

Після того як потік (будь то зображення, відео або потік з камери) було додано на слайд, програма додає цей потік до списку потоків в інтерфейсі користувача. Список відображається в елементі ListBox, де кожен елемент містить назву потоку та джерело. Цей список є важливим елементом управління, оскільки дозволяє користувачеві бачити всі активні потоки та взаємодіяти з ними. Користувач може вибрати потік зі списку, щоб він став активним і відображався на відповідному слайді.

Завдяки такій реалізації користувач має можливість не лише додавати різні типи медіа на слайди, але й керувати ними через інтерфейс програми. Кожен потік, будь то зображення, відео або камера, може бути легко доданий і відображений у вікні програми без потреби вручну налаштовувати медіа-потоки. Програма динамічно оновлюється, що дозволяє забезпечити інтерактивний процес редагування та управління мультимедійними елементами в реальному часі.

Таким чином, функція AddStreamBTN_Click не лише реалізує додавання нових потоків, але й забезпечує динамічну взаємодію з медіа-ресурсами, дозволяючи створювати гнучкий і зручний інтерфейс для роботи з мультимедійними елементами в межах програми.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Ручне тестування програмного забезпечення

Ручне тестування було проведено для детального аналізу роботи програмного забезпечення та оцінки його відповідності вимогам, поставленим під час розробки. У тестуванні брали участь чотири користувачі, які мали різний рівень технічної підготовки: від початківців до досвідчених спеціалістів у галузі роботи з мультимедійними системами. Такий підхід дозволив оцінити роботу програми з різних точок зору та виявити як технічні аспекти використання.

Перший етап тестування включав перевірку базових функцій програми. Користувачі створювали нові слайди, заповнювали їх текстовими елементами, зображеннями та відеофайлами, а також налаштовували параметри відображення, такі як розмір, колір і позиція об'єктів. Особлива увага приділялася перевірці інтерактивних елементів інтерфейсу, таких як панелі налаштувань та меню.

На другому етапі тестували можливість інтеграції програми з зовнішніми пристроями. Користувачі підключали до програми камери, мікрофони та суфлери, перевіряючи, наскільки швидко і стабільно програма розпізнає пристрої та передає дані в реальному часі. Було виявлено, що камера і мікрофон працюють стабільно, але при використанні живого відео на екрані іноді з'являлися затримки та незначні спотворення зображення.

Третій етап був присвячений тестуванню дизайну слайдів. Користувачі випробовували вбудовані шаблони, налаштовували кольорові схеми, застосовували градієнти та перевіряли, як змінюється вигляд слайдів у різних конфігураціях. Були виявлені певні незручності у використанні панелі налаштувань, оскільки вона вимагала багато кліків для доступу до потрібних функцій, що значно ускладнювало роботу для початківців.

Завершальний етап включав перевірку роботи програми в умовах реального навантаження. Користувачі створювали презентації, що містили велику кількість слайдів із різними потоками даних, та запускали їх у режимі трансляції. У деяких випадках виникали труднощі із синхронізацією звуку та відео, а також знижувалася швидкість реагування інтерфейсу, особливо під час редагування великих слайдів.

Загалом ручне тестування дозволило виявити низку технічних і функціональних проблем, а також зібрати корисні відгуки про зручність використання програми. Користувачі відзначили інтуїтивно зрозумілий інтерфейс, але вказали на необхідність оптимізації роботи з великими проєктами та спрощення доступу до деяких функцій. На основі отриманих результатів було

складено список рекомендацій для вдосконалення програми, що включав оптимізацію алгоритмів обробки даних, покращення дизайну панелей інструментів і спрощення взаємодії з інтерактивними елементами.

1.2 Автоматизоване тестування програмного забезпечення

Автоматизоване тестування програми було виконано з метою перевірки стабільності та коректності роботи окремих модулів, а також виявлення потенційних проблем, які могли залишитися непоміченими під час ручного тестування. Для цього було розроблено набір unit-тестів, що охоплювали основні функціональні частини програми, зокрема обробку даних, збереження і завантаження проєктів, роботу з потоками даних та їх трансляцію. Тестування проводилося із використанням фреймворку NUnit, який забезпечував зручне виконання тестів, а також автоматичний збір і аналіз результатів.

На першому етапі автоматизованого тестування були перевірені модулі обробки даних. Тести включали симуляцію додавання текстових, графічних, аудіо- та відеопотоків на слайди, їхнє переміщення, зміну параметрів і видалення. Тести показали, що базова функціональність виконується коректно, однак у певних сценаріях виникали проблеми з обробкою нестандартних відеоформатів, що призводило до помилок при завантаженні файлів.

Другий етап включав перевірку збереження та завантаження проєктів. Для цього було створено кілька тестових презентацій із різними наборами даних і конфігурацій. Тести перевіряли, чи правильно зберігаються всі параметри слайдів і потоків, а також чи коректно відновлюється структура проєкту після завантаження. У більшості випадків модуль збереження працював стабільно, однак виявлено, що деякі особливі символи у назвах файлів спричиняли помилки під час відновлення проєктів.

Третій етап тестування був зосереджений на продуктивності програми при високих навантаженнях. Було створено сценарії, що симулювали одночасну роботу

з великою кількістю слайдів і потоків даних, включаючи живі трансляції з камери та мікрофона. Тести показали, що програма починає втрачати стабільність при значному збільшенні кількості потоків, що призводило до затримок у відображенні відео та зниження швидкості інтерфейсу.

На завершальному етапі проводилося тестування стійкості програми до помилкових дій користувача, таких як неправильний формат вхідних даних або відключення зовнішніх пристроїв під час роботи. Тести підтвердили, що програма здатна коректно обробляти більшість таких ситуацій, однак були виявлені окремі випадки, коли програма припиняла роботу без належного повідомлення про помилку.

Результати автоматизованого тестування показали, що основна функціональність програми відповідає вимогам, але є аспекти, які потребують подальшого вдосконалення. До них належать оптимізація обробки нестандартних форматів файлів, підвищення продуктивності при роботі з великими навантаженнями та покращення механізмів обробки помилок. Автоматизовані тести також надали цінну інформацію про можливі сценарії використання програми, що дозволить зосередитися на пріоритетних напрямках розвитку.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Комерційний та технологічний аудит науково-технічної розробки

Метою цього розділу є проведення технологічного аудиту програмного забезпечення [27] для здійснення інтерактивних багатоканальних відеотрансляцій. Це програмне забезпечення призначене для використання в різноманітних сценаріях, включаючи інтерактивні зустрічі, медіа-презентації, освітні сесії, а також для організації онлайн-конференцій. Основною особливістю проекту є його здатність об'єднувати різні канали вхідних даних, такі як відео, аудіо, текст, зображення, і дозволяти користувачам керувати цими потоками в реальному часі для створення інтерактивного і динамічного вмісту.

Проект можна розглядати як універсальне рішення для створення медіа-додатків, що забезпечує гнучкість у налаштуванні і настройці потоків даних. Подібно до популярних інструментів, таких як Zoom або OBS, це програмне забезпечення дозволяє користувачам керувати джерелами відео і аудіо, інтегрувати різні формати даних і створювати персоналізовані трансляції. Порівняно з рішеннями на ринку, яке надається іншими розробниками, дане програмне забезпечення пропонує більш гнучкі можливості налаштування потоків, з можливістю інтеграції додаткових джерел даних, таких як камери, мікрофони, або мультимедійні файли.

Аналогом можуть бути популярні платформи, такі як Zoom, OBS або Microsoft Teams. Однак, порівняно з ними, дане програмне забезпечення виділяється своєю здатністю обробляти більш складні сценарії з'єднань, надаючи розширені функції для обробки різних типів даних, включаючи обробку зображень, інтерактивні елементи, а також можливість об'єднання різних джерел в один поток. З урахуванням цих додаткових можливостей, комерційний потенціал проекту значно вищий, ніж у конкурентів, що пропонують базові функції відео- і аудіо трансляцій.

Для проведення комерційного та технологічного аудиту до процесу залучають не менше трьох незалежних експертів. Оцінка науково-технічного рівня розробки та її комерційного потенціалу [28] рекомендована для проведення за п’ятибальною системою оцінювання, яка базується на 12 критеріях. Це дозволяє забезпечити об'єктивність оцінки програмного забезпечення з точки зору його технічних характеристик, зручності використання, гнучкості і потенціалу на ринку. Включення цих факторів у оцінку допоможе створити повну картину потенційної конкурентоспроможності проекту.

Таблиця 3 — Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за 5-ти бальною шкалою)					
Критерій	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів

Продовження таблиці 3

Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практик на здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Усі дані по кожному параметру занесено в таблиці 4

Таблиця 4 — Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	ПІБ експертів		
	Експерт 1	Експерт 2	Експерт 3
	Бали		
Технічна здійсненність концепції	4	3	4
Наявність аналогів на ринку	4	4	4
Цінова політика	3	3	4
Технічні та споживчі властивості виробу	4	3	4
Експлуатаційні витрати	4	4	3
Ринок збуту	3	3	4
Конкурентоспроможність	3	4	4
Фахівці з технічної і комерційної реалізації	4	3	4
Фінансування	4	4	4
Матеріально-технічна база	3	3	4
Термін реалізації ідеї	3	4	4
Супровідна документація	4	3	4
Сума	43	41	47
Середньоарифметична сума балів	(43+41+47) / 3 = 43.6		

За даними таблиці 4 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 5.

Таблиця 5 — Рівні комерційного потенціалу розробки

Середньоарифметична сума балів, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

Як видно з таблиці, рівень комерційного потенціалу розроблюваного нового програмного продукту є високим, що досягається за рахунок автоматизація процесів взаємодії всіх потоків даних в межах єдиної системи та аналізу відповідних алгоритмів.

5.2 Прогнозування витрат на виконання науково дослідної (дослідно-конструкторської) роботи

5.2.1 Основна заробітна плата розробників

Основна заробітна плата розробників розраховується за формулою:

$$З_о = \frac{М}{Т_р} \cdot t, \quad (5.1)$$

де М — місячний посадовий оклад конкретного розробника (дослідника), грн.;

$Т_р$ — число робочих днів за місяць, 22 днів;

t — число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 6.

Оскільки в даному випадку розробляється програмний продукт, то розробник виступає одночасно і основним робітником, і тестувальником розроблюваного програмного продукту.

Таблиця 6 — Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	32500	1477,27	32	47272,727
Програміст	30000	1363,64	32	43636,364
Всього				90909,09

5.2.2 Додаткова Заробітна плата розробників

Додаткову заробітну плату прийнято розраховувати як 14,3 % від основної заробітної плати розробників та робітників:

$$З_д = З_о \cdot 14,3\ \% / 100\ \% \quad (5.2)$$

$$З_д = (90909,09 \cdot 14,3\ \% / 100\ \%) = 13000,00 \ \text{(грн.)}$$

5.2.3 Нарахування на заробітну плату розробників.

Згідно з діючим законодавством нарахування на заробітну плату складають 22 % від суми основної та додаткової заробітної плати.

$$Н_з = (З_о + З_д) \cdot 22\ \% / 100\% \quad (5.3)$$

$$Н_з = (90909,09 + 13000,00) \cdot 22\ \% / 100\ \% = 22860,00 \ \text{(грн.)}$$

5.2.4. Витрати на меиеріали та комплектуючі

Оскільки для розроблювального програмного продукту не потрібно витрачати матеріали та комплектуючи, то витрати на матеріали і комплектуючі дорівнюють нулю.

5.2.5 Амортизація обладнання

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді розраховується за формулою:

$$A = \frac{Ц}{Тв} \cdot \frac{t_{вик}}{12} \text{ [грн.].} \quad (5.4)$$

де Ц — балансова вартість обладнання, грн.;

Т — термін корисного використання обладнання згідно податкового законодавства, років

$t_{вик}$ — термін використання під час розробки, місяців

Розрахуємо, для прикладу, амортизаційні витрати на комп'ютер балансова вартість якого становить 20000 грн., термін його корисного використання згідно податкового законодавства – 2 роки, а термін його фактичного використання – 1,45 міс.

$$A_{обл} = \frac{20000}{2} \times \frac{1{,}45}{12} = 1212{,}12 \; грн.$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до таблиці 7. Так як вартість ліцензійної ОС та спеціалізованих ліцензійних нематеріальних ресурсів менше 20000 грн, то даний нематеріальний актив не амортизується, а його вартість включається у вартість розробки повністю, $B_{нем.ак.}$ = 8000 грн.

Таблиця 7 — Амортизаційні відрахування на матеріальні та нематеріальні ресурси для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортиза-ційні відрахуван-ня, грн.
Комп'ютер та комп'ютерна периферія	20000	2	1,45	1212,121
Офісна техніка (монітори, колонки)	28000	4	1,45	848,485
Приміщення	1200000	20	1,45	7272,727
Всього				9333,33

5.2.6 Витрати на електроенергію

Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. При цьому тарифи на розподіл електроенергії у різних постачальників (енергорозподільних компаній), будуть різними. Крім того, розмір тарифу залежить від класу напруги (1-й або 2-й клас). Тарифи на розподіл електроенергії для всіх енергорозподільних компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п, \quad (5.5)$$

де В — вартість 1 кВт-години електроенергії для 1 класу підприємства з ПДВ в 2024 році для Вінницької області за даними Енера-Вінниця, В = 10,42 грн./кВт;

П — встановлена потужність обладнання, кВт. П = 0,3 кВт;

Ф — фактична кількість годин роботи обладнання, годин.

$К_п$ — коефіцієнт використання потужності, $К_п$ = 0,9.

$$В_е = 0,9 \cdot 0,3 \cdot 8 \cdot 32 \cdot 10,42 = 720,2304 \text{ (грн.)}$$

5.2.7 Інші витрати та загальновиробничі витрати.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.6)$$

де $Н_{ів}$ — норма нарахування за статтею «Інші витрати».

$$I_в = 90909,09 * 80\% / 100\% = 72727,27 \text{ (грн.)}$$

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$Н_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.7)$$

де $Н_{нзв}$ — норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$$Н_{нзв} = 90909,09 * 130\ \% / 100\ \% = 118182 \text{ (грн.)}$$

5.2.8 Витрати на проведення науково-дослідної роботи.

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

$$В_{заг} = 90909,09+13000,00+22860,00+9333,33+8000+720,23+72727,27+$$
$$+118182 = 335731,75 \text{ грн.}$$

5.2.9 Розрахунок загальних витрат на науково-дослідну (науково-технічну) роботу та оформлення її результатів.

Загальні витрати на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} \text{ (грн)}, \quad (5.8)$$

де η — коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η=0,1; технічного проектування, то η=0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7; впровадження, то η=0,9. Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка:

$$ЗВ = 335731{,}75 / 0{,}5 = 671463 \text{ грн.}$$

5.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнювальним позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Для того, щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від можливого впровадження науково-технічної розробки необхідно:

— вказати, з якого часу можуть бути впроваджені результати науково-технічної розробки;

— зазначити, протягом скількох років після впровадження цієї науково-технічної розробки очікуються основні позитивні результати для потенційного інвестора (наприклад, протягом 3-х років після її впровадження);

— кількісно оцінити величину існуючого та майбутнього попиту на цю або аналогічні чи подібні науково-технічні розробки та назвати основних суб’єктів (зацікавлених осіб) цього попиту;

— визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями.

При розрахунку економічної ефективності потрібно обов’язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу. При оцінюванні ефективності інноваційних проектів передбачається розрахунок таких важливих показників:

— абсолютного економічного ефекту (чистого дисконтованого доходу);

— внутрішньої економічної дохідності (внутрішньої норми дохідності);

— терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково-технічних розробок, розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором можна об’єднати, враховуючи визначені ситуації з відповідними умовами.

5.3.1 Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.9)$$

де $\pm\Delta Ц_o$ — зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу;

N — кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки;

$Ц_o$ — основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_o = Ц_б \pm \Delta Ц_o$;

$Цб$ — вартість програмного продукту у році до впровадження результатів розробки;

ΔN — збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

λ — коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт $\lambda = 0{,}8333$.

p — коефіцієнт, який враховує рентабельність продукту;

ϑ — ставка податку на прибуток, у 2024 році ϑ =18%.

Припустимо, що при прогнозованій ціні 8200 грн. за одну копію, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на *500* грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року — на 3000 шт., протягом другого року — на 2000 шт., протягом третього року на 1000 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

$\Delta П_1$ = (0*500 + (8200 + 500)*3000)* 0,8333* 0,27) * (1 - 0,18) = 4538699,818 грн.

$\Delta П_2$ = (0*500 + (8200 + 500)*(3000+2000)* 0,8333* 0,27) * (1 - 0,18) = 8025749,679 грн.

$\Delta П_3$ = (0*500 + (8200 + 500)*(3000+2000+1000)* 0,8333* 0,27) * (1 - 0,18) = 9630899,615 грн.

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 22195349,11 грн.

5.3.2 Розрахунок ефективності вкладених інвестицій та періоду їх окупності.

Розраховуємо приведену вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_1^T \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.10)$$

де $\Delta П_i$— збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної (науково-технічної) роботи, грн;

Т — період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки;

τ— ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0,05...0,15$;

t — період часу (в роках).

Збільшення прибутку ми отримаємо, починаючи з першого року:

ПП = (4538699,818/(1+0,1)[1])+(8025749,679/(1+0,1)[2])+(9630899,615/
/(1+0,1)[3]) = 4126090,74 + 6632850,974 + 7235837,427 = 17994779,14 грн.

Далі розраховують величину початкових інвестицій *PV*, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} * ЗВ, \quad (5.11)$$

де $k_{інв}$ — коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}=2...5$, але може бути і більшим;

ЗВ — загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

PV = 2 * 671463 = 1342926,98 *грн.*

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід *(NPV, Net Present Value)* для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП – PV, \quad (5.12)$$

$E_{абс}$ = 17994779,14 – 1342926,98 = 16651852,16 грн.

Оскільки $E_{абс} > 0$то вкладання коштів на виконання та впровадження результатів даної науково-дослідної (науково-технічної) роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розра-хувати внутрішню економічну дохідність або показник внутрішньої но-рми дохідності *(IRR, Internal Rate of Return)* вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій$E_в$. Для цього використаємо формулу:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.13}$$

$T_ж$— життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{(1 + 16651852{,}16/1342926{,}98} - 1 = 1{,}375$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \tag{5.14}$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні d = (0,09…0,14);

f— показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

$$\tau_{min}.$$

Так як $E_в > \tau_{min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в}, \quad (5.15)$$

$$T_{ок} = 1 / 1{,}375 = 0{,}73 \text{ р.}$$

Оскільки $T_{ок}$ < 3-х років, а саме термін окупності рівний 0,73 роки, то фінансування даної наукової розробки є доцільним.

Висновки до розділу: економічна частина даної роботи містить розрахунок витрат на розробку нового програмного продукту, сума яких складає 671463 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є висококонкурентоспроможним. Період окупності складе близько 0,73 роки.

ВИСНОВКИ

У роботі було розглянуто сучасний стан розробки програмного забезпечення для проведення інтерактивних багатоканальних відеотрансляцій, а також проаналізовано основні виклики, які постають перед розробниками таких продуктів. Було детально досліджено інструменти та технології, які зазвичай використовуються в розробці таких систем, зокрема в контексті обробки відео, аудіо, зображень та інших мультимедійних даних в реальному часі. Проведений аналіз показав необхідність створення програмного продукту, який дозволяє ефективно обробляти різні типи вхідних даних і забезпечує гнучкість у їх використанні в рамках інтерактивних трансляцій.

У другому розділі було розроблено архітектуру програмного забезпечення для проведення багатоканальних відеотрансляцій. Вибір алгоритмів і технологій для реалізації таких функцій, як обробка різних типів медіа-даних, організація потоків і синхронізація з різними джерелами, був обґрунтований з огляду на вимоги до надійності, гнучкості та масштабованості системи. Було розроблено моделі для інтеграції відео, аудіо та графічних елементів, а також визначено методи оптимізації використання апаратних ресурсів.

У третьому розділі було обґрунтовано вибір технології програмування та інструментів для реалізації програмного забезпечення. Для цього була обрана об'єктно-орієнтована мова програмування C# у поєднанні з платформою WPF, яка забезпечує потужні можливості для створення графічних інтерфейсів користувача та обробки різноманітних мультимедійних потоків. Також було проведено тестування програмного продукту, порівняно цифрові показники з аналогами, що підтвердило високу ефективність розробленої системи в порівнянні з конкурентами.

У четвертому розділі було проведено комерційний та технологічний аудит розробленого програмного забезпечення, а також здійснено аналіз економічної

ефективності його впровадження. Результати аудиту показали високий рівень конкурентоспроможності продукту, а також швидкий період окупності інвестицій завдяки ефективності використання ресурсів і можливості інтеграції з іншими платформами.

Таким чином, мета роботи — розробка програмного забезпечення для інтерактивних багатоканальних відеотрансляцій з високою гнучкістю в обробці різних типів медіа-даних та забезпеченням ефективного використання ресурсів — була досягнута, що забезпечує значний крок у напрямку розвитку технологій трансляцій і мультимедійних інтерфейсів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Л. А. Савицька, Д. В. Куклій, Н. В. Добровольська. СТРІМІНГОВІ ПЛАТФОРМИ ТА ЇХ РОЛЬ У РОЗВИТКУ МУЛЬТИМЕДІЙНОЇ КОМУНІКАЦІЇ // [Електронний ресурс] / Л. А. Савицька // Матеріали LIV Всеукраїнської науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії, Вінниця, 24А27 березня 2025 р. – Електрон. текст. дані. – 2024. – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/paper/view/22729

2. Пашков, Р. А. Роздільна здатність приладів тепловізорів за критеріями Джонсона / Р. А. Пашков, Н. О. Балахонова // XII Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених «Погляд у майбутнє приладобудування», 15-16 травня 2019 р., м. Київ, Україна : збірник праць / КПІ ім. Ігоря Сікорського, ПБФ. – Київ : КПІ ім. Ігоря Сікорського, 2019. – С. 117–120. – Бібліогр.: 5 назв.

3. Зеленько, І. Р. Збільшення частоти кадрів відео використовуючи методи штучного інтелекту : дипломна робота ... бакалавра : 122 Комп'ютерні науки / Зеленько Ілля Русланович. – Київ, 2024. – 68 с.

4. Марин, Р. А. Система автофокусування відео- та фотокамер на рухомих конструкціях : дипломний проєкт ... бакалавра : 151 Автоматизація та комп'ютерно-інтегровані технології / Марин Рувім Андрійович. – Київ, 2023. – 72 с.

5. Сав`юк, В. С. Автоматизована система стабілізації відеокамери : магістерська дис. : 151 Автоматизація та комп'ютерно-інтегровані технології / Сав`юк Володимир Сергійович. – Київ, 2024. – 96 с.

6. Паляниця, Ю. Б. "Спосіб виділення акустичного сигналу на тлі завад." Матеріали IV Всеукраїнської студентської науково-технічної

конференції „Природничі та гуманітарні науки. Актуальні питання “ 1 (2011): 296-296.

7. Ларіонов, Я. О. Дослідження апаратно-програмних засобів для відеоблогінгу : дипломний проєкт ... бакалавра : 171 Електроніка / Ларіонов Ярослав Олександрович. – Київ, 2023. – 75 с.

8. Тарасенко, Г. В. (2022). Електронна система мікшування аудиосигналів з використанням Bluetoots-каналів (Master's thesis, Сумський державний університет).

9. Shah, A. D., Agrawal, S., & Sharma, K. (2015). Transport Stream Playout System for MPEG-TS using Program Clock Reference. International Journal of Computer Applications, 117(16), 22-25

10. Mills, D. L. (2003). A brief history of NTP time: Memoirs of an Internet timekeeper. ACM SIGCOMM Computer Communication Review, 33(2), 9-21.

11. Watt, Steve T., et al. "Understanding and applying precision time protocol." 2015 Saudi Arabia Smart Grid (SASG). IEEE, 2015.

12. Вернудін, А. І. Дослідження особливостей мастерингу музичних для стрімінгових платформ : магістерська дис. : 171 Електроніка / Вернудін Антон Ігорович. – Київ, 2024. – 77 с.

13. Vavilenkova, Anastasiia. "Аналіз гнучких методологій розробки програмного забезпечення для реалізації у командних проєктах." Вісник Національного технічного університету «ХПІ». Серія: Нові рішення у сучасних технологіях 1 (7) (2021): 39-46.

14. Vintenko, Boris, et al. "ДОСЛІДЖЕННЯ НОРМАТИВНИХ ДОКУМЕНТІВ ТА ГАЛУЗЕВИХ СТАНДАРТІВ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ УПРАВЛІННЯ АЕС, ВАЖЛИВИХ ДЛЯ БЕЗПЕКИ." Системи управління, навігації та зв’язку. Збірник наукових праць 2.72 (2023): 170-178.

15. Руда, О. А., and Ю. Б. Моденов. "Методи та моделі тестування програмного забезпечення." Problems of Informatization and Control 2.42 (2013): 93-98.

16. Святненко, В., and І. Нетреба. "Практичні аспекти впровадження інформаційних систем управління на вітчизняних підприємствах." Вісник Київського національного університету імені Тараса Шевченка. Економіка 137 (2012): 26-31.

17. Равлюк, Максим, and Йосиф Ситник. "ОПТИМІЗАЦІЯ AGILE: СИНЕРГІЯ SCRUM І KANBAN У РОЗРОБЦІ ТА ПІДТРИМЦІ ПРОГРАМНОГО ПРОДУКТУ." Економіка та суспільство 63 (2024).

18. Amann, Sven, et al. "A study of visual studio usage in practice." 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER). Vol. 1. IEEE, 2016.

19. Chappell, David, and David Wayne Chappell. "Understanding. NET: a tutorial and analysis." (2002).

20. Hejlsberg, Anders, Scott Wiltamuth, and Peter Golde. C# language specification. Addison-Wesley Longman Publishing Co., Inc., 2003.

21. Sells, Chris, and Ian Griffiths. Programming WPF: Building Windows UI with Windows Presentation Foundation. " O'Reilly Media, Inc.", 2007.

22. Ghoda, Ashish, and Mamta Dalal. XAML developer reference. Pearson Education, 2011

23. Hutchens, David H., and Victor R. Basili. "System structure analysis: Clustering with data bindings." IEEE transactions on Software Engineering 8 (1985): 749-757.

24. Law, Effie Lai-Chong, Paul Van Schaik, and Virpi Roto. "Attitudes towards user experience (UX) measurement." International Journal of Human-Computer Studies 72.6 (2014): 526-541.

25. Wilson, Andrew, and Steven Shafer. "XWand: UI for intelligent spaces." Proceedings of the SIGCHI conference on Human factors in computing systems. 2003.

26. MacVittie, Lori A. XAML in a Nutshell. " O'Reilly Media, Inc.", 2006.

27. Цибинога, М. О. "Технологічний аудит як метод оцінки результатів науково-технічних проектів." (2010).

28. Кобєлєва, Анна. "Ранжування інноваційних технологій за критерієм рівня готовності технологій до комерціалізації та її комерційного потенціалу." Вісник Національного технічного університету" Харківський політехнічний інститут"(економічні науки) 2 (2024): 52-58.

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

проф., д.т.н.. Азаров О.Д.

_____

" ____ " _____ 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

"Програмно-апаратні засоби для проведення інтерактивних багатоканальних відеотрансляцій"

08-54.МКР.010.00.000 ТЗ

Керівник роботи к.т.н. доц. каф. ОТ

_____ Савицька Л. А.

Студент групи 1КІ–23м

_____ Куклій Д. В.

ВНТУ 2024

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Розробка програми необхідна для підвищення якості й ефективності проведення відеотрансляцій необхідно розробити програмний продукт для зручної інтеграції різних медіапотоків в режимі реального часу

1.2 Наказ про затвердження теми МКР

2 Мета МКР і призначення розробки

2.1 Мета роботи - створення програмного забезпечення, яке дасть можливість інтегрувати в єдину систему різні медіапотоки та пристрої, а також ефективно ними керувати в режимі реального часу.

2.2 Призначення розробки – створення програмного забезпечення, яке дасть можливість об’єднати різні медіапотоки та синхронізувати їх.

3 Вихідні дані для виконання МКР

3.1 Призначення системи – проведення інтерактивних багатоканальних відеотрансляцій.

3.2 Використовувані інструменти – комп’ютер, пристрої вводу, пристрої виводу.

4 Вимоги до виконання МКР

4.1 Провести аналіз існуючих апаратних та програмних засобів.

4.2 Розробити механізми для синхронізації медіапотоків.

4.3. Провести тестування ефективності та надійності системи.

4.4. Оцінити потенціал системи для її комерціалізації

5 Етапи МКР та очікувані результати

Етапи та очікувані результати наведено у таблиці А.1

Таблиця А.1 – Етапи та очікувані результати МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		Початок	Кінець	
1	Аналіз існуючих технологій, огляд апаратних аналогів	19.09.2024	15.10.2024.	Вступ, Розділ 1
2	Огляд програмних аналогів	16.10.2024	29.10.2024	Розділ 2
3	Розробка структурної схеми	29.10.2024	05.11.2024	Розділ 3
4	Розробка дизайну	06.11.2024	12.11.2024	Розділ 4
5	Розробка основного функціоналу	12.11.2024	25.11.2024	Розділ 4
6	Підготовка економічної частини	25.11.2024	03.12.2024	Розділ 5
7	Оформлення пояснювальної записки і графічного матеріалу презентації	04.12.2024	10.12.2024	ПЗ, графічний матеріал у презентації

6 Матеріали, що подаються до захисту МКР

До захисту подаються: пояснювальна записка МКР, графічні і ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відгук наукового керівника, відгук опонента, анотації до МКР українською та іноземною мовами.

7 Порядок контролю виконання та захисту МКР

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

8. Вимоги до оформлення та порядок виконання МКР

8.1 При оформлюванні МКР 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 «Єдина система конструкторської документації. Основні написи»;

— методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія»;

— документи на які посилаються у вище вказаних.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ–03.02.02 П.001.01:21

ДОДАТОК Б

Лістинг файлу MainWindow.xaml

```
<Window x:Class="MultiStream.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
        mc:Ignorable="d"
        Title="MultiStream"
        WindowState="Maximized"
        WindowStyle="SingleBorderWindow"
        Background="#1E1E1E">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="25*" />
            <ColumnDefinition Width="75*" />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="50*" />
            <RowDefinition Height="50*" />
        </Grid.RowDefinitions>
        <!-- Ліва верхня частина: Список слайдів і кнопка -->
        <StackPanel Grid.Column="0" Grid.Row="0" Background="#2C2C2C"
Margin="5">
            <ListBox x:Name="SlideList"
                     Foreground="White"
                     Background="#333333"
                     BorderThickness="1"
```

```
                BorderBrush="#444444"
                VerticalAlignment="Top"
                Height="290"
                ScrollViewer.VerticalScrollBarVisibility="Auto"
                SelectionChanged="SlideList_SelectionChanged"
                PreviewKeyDown="SlideList_PreviewKeyDown">
            <!-- Шаблон елемента списку -->
            <ListBox.ItemTemplate>
              <DataTemplate>
                <StackPanel>
                  <TextBlock x:Name="SlideName" Foreground="White"
Padding="5" />
                </StackPanel>
              </DataTemplate>
            </ListBox.ItemTemplate>
          </ListBox>
          <Button x:Name="AddSlideBTN"
                Content="Додати слайд"
                Foreground="White"
                Background="#444444"
                Margin="5"
                Padding="10"
                HorizontalAlignment="Stretch"
                Click="AddSlideBTN_Click" />
        </StackPanel>
        <!-- Ліва нижня частина: Список потоків і кнопка -->
```

```
            <StackPanel Grid.Column="0" Grid.Row="1" Background="#2C2C2C"
Margin="5">
                <ListBox x:Name="StreamList"
                Foreground="White"
                Background="#333333"
                BorderThickness="1"
                BorderBrush="#444444"
                VerticalAlignment="Top"
                Height="290"
                ScrollViewer.VerticalScrollBarVisibility="Auto" />
                <Button x:Name="AddStreamBTN"
                        Content="Додати потік"
                        Foreground="White"
                        Background="#444444"
                        Margin="5"
                        Padding="10"
                        HorizontalAlignment="Stretch"
                        />
            </StackPanel>
            <!-- Права частина: Поле для відображення слайда -->
            <Grid Grid.Column="1" Grid.RowSpan="2" Background="#1E1E1E"
Margin="5">
                <Border Background="#2C2C2C"
                        BorderThickness="1"
                        BorderBrush="#444444"
                        HorizontalAlignment="Center"
                        VerticalAlignment="Center"
```

```
                Width="800"
                Height="450">
            <Canvas x:Name="slideCanvas"
                    Background="White"
                    Visibility="Collapsed" />
        </Border>
    </Grid>
  </Grid>
</Window>
```

ДОДАТОК В

Лістинг файлу MainWindow.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
namespace MultiStream
{
   public partial class MainWindow : Window
   {
      private int _slideCounter = 0;
      private Dictionary<string, Canvas> _slides = new Dictionary<string, Canvas>();
      public MainWindow()
      {
         InitializeComponent();
      }
      private void AddSlideBTN_Click(object sender, RoutedEventArgs e)
      {
         _slideCounter++;
         string slideName = $"Слайд {_slideCounter}";
         // Створення нового Canvas для слайда
         Canvas newSlideCanvas = new Canvas
         {
```

```
            Background = Brushes.White,
            Width = 800,  // Встановлення розміру канвасу
            Height = 450  // Встановлення розміру канвасу
        };
        _slides[slideName] = newSlideCanvas;
        // Додавання нового елемента до списку
        ListBoxItem listBoxItem = new ListBoxItem
        {
            Content = slideName,
            Foreground = Brushes.White
        };
        // Подія для редагування назви слайда
        listBoxItem.MouseDoubleClick += (s, args) =>
        {
            EditSlideName(listBoxItem);
        };
        SlideList.Items.Add(listBoxItem);
        SlideList.SelectedItem = listBoxItem; // Автовибір нового слайда
    }
    private void SlideList_SelectionChanged(object sender,
SelectionChangedEventArgs e)
    {
        if (SlideList.SelectedItem is ListBoxItem selectedItem)
        {
            string slideName = selectedItem.Content.ToString();
            if (_slides.TryGetValue(slideName, out Canvas slideCanvas))
            {
```

```
            // Відображення вибраного слайда
            slideCanvas.Children.Clear();
            slideCanvas.Children.Add(slideCanvas);

            // Зробити canvas видимим
            slideCanvas.Visibility = Visibility.Visible;
        }
    }
    else
    {
        // Якщо слайд не вибрано, приховуємо Canvas
        slideCanvas.Children.Clear();
        slideCanvas.Visibility = Visibility.Collapsed;
    }
}
private void EditSlideName(ListBoxItem item)
{
    string oldName = item.Content.ToString();

    // Перевірка, чи слайд існує
    if (!_slides.ContainsKey(oldName))
        return;
    // Створення текстового поля для редагування
    TextBox textBox = new TextBox
    {
        Text = oldName,
        Foreground = Brushes.White,
```

```
                Background = Brushes.Black,
                BorderBrush = Brushes.Gray,
                BorderThickness = new Thickness(1),
                Padding = new Thickness(2)
            };
            textBox.LostFocus += (s, args) =>
            {
                string newName = textBox.Text;
                // Оновлення назви слайда
                if (!string.IsNullOrWhiteSpace(newName) &&
!_slides.ContainsKey(newName))
                {
                    _slides[newName] = _slides[oldName];
                    _slides.Remove(oldName);
                    item.Content = newName;
                }
                else
                {
                    item.Content = oldName;
                }
                SlideList.Items.Refresh();
            };
            textBox.KeyDown += (s, args) =>
            {
                if (args.Key == Key.Enter || args.Key == Key.Return)
                {
```

```
                textBox.MoveFocus(new
TraversalRequest(FocusNavigationDirection.Next));
            }
        };
        item.Content = textBox;
        textBox.Focus();
        textBox.SelectAll();
    }
    private void AddStreamBTN_Click(object sender, RoutedEventArgs e)
    {
        var addStreamWindow = new AddStreamWindow();
        addStreamWindow.Owner = this;
        if (addStreamWindow.ShowDialog() == true)
        {
            string streamName = addStreamWindow.StreamName;
            string source = addStreamWindow.SelectedSource;
            // Додати у список потоків
            ListBoxItem listBoxItem = new ListBoxItem
            {
                Content = $"{streamName} ({source})",
                Foreground = Brushes.White
            };
            StreamList.Items.Add(listBoxItem);
            // Логіка для подальшого використання потоку (зображення,
відео, камера)
            if (source.EndsWith(".jpg") || source.EndsWith(".png") ||
source.EndsWith(".bmp"))
```

```
            {
                // Логіка для зображень
                BitmapImage bitmap = new BitmapImage(new Uri(source)); // Завантажуємо зображення з файлу
                // Створення Image для відображення на слайді
                Image image = new Image
                {
                    Source = bitmap,
                    Width = 800,  // Встановлення розміру зображення
                    Height = 450  // Встановлення розміру зображення
                };
                // Визначаємо слайд (наприклад, перший)
                string slideName = $"Слайд {_slideCounter}";
                if (_slides.ContainsKey(slideName))
                {
                    Canvas slideCanvas = _slides[slideName];
                    slideCanvas.Children.Clear(); // Очищаємо попередній вміст
                    slideCanvas.Children.Add(image);  // Додаємо нове зображення
                }
            }
            else if (source.EndsWith(".mp4") || source.EndsWith(".avi"))
            {
                // Логіка для відео
                MediaElement video = new MediaElement
                {
                    Source = new Uri(source),  // Встановлення джерела відео
```

```
                    Width = 800,  // Встановлення розміру відео
                    Height = 450,  // Встановлення розміру відео
                    LoadedBehavior = MediaState.Manual,  // Автозапуск відео
                    UnloadedBehavior = MediaState.Manual
                };
                // Визначаємо слайд (наприклад, перший)
                string slideName = $"Слайд {_slideCounter}";
                if (_slides.ContainsKey(slideName))
                {
                    Canvas slideCanvas = _slides[slideName];
                    slideCanvas.Children.Clear(); // Очищаємо попередній вміст
                    slideCanvas.Children.Add(video); // Додаємо відео на слайд
                    video.Play();  // Запускаємо відео
                }
            }
            else
            {
                // Логіка для камери
                // Створення нового об'єкта VideoCapture для отримання відео
з камери (наприклад, за допомогою бібліотеки OpenCV або MediaCapture)
                MediaCapture mediaCapture = new MediaCapture();
                mediaCapture.InitializeAsync().ContinueWith(task =>
                {
                    if (task.Status == TaskStatus.RanToCompletion)
                    {
                        // Отримуємо відео з камери
                        var preview = new CaptureElement
```

```
                    {
                        Width = 800,     // Встановлення розміру для
відображення відео
                        Height = 450
                    };
                    // Прив'язуємо CaptureElement до MediaCapture для
відображення відео
                    mediaCapture.SetPreviewElement(preview);
                    Dispatcher.Invoke(() =>
                    {
                        // Визначаємо слайд (наприклад, перший)
                        string slideName = $"Слайд {_slideCounter}";
                        if (_slides.ContainsKey(slideName))
                        {
                            Canvas slideCanvas = _slides[slideName];
                            slideCanvas.Children.Clear();         //         Очищаємо
попередній вміст
                            slideCanvas.Children.Add(preview); // Додаємо відео
з камери
                        }
                    });
                    mediaCapture.StartPreviewAsync();
                }
            });
        }
    }
}
```

```
// Обробка натискання клавіші Delete
private void SlideList_PreviewKeyDown(object sender, KeyEventArgs
e)
{
   if (e.Key == Key.Delete)
   {
      // Перевірка, чи є вибраний елемент
      if (SlideList.SelectedItem is ListBoxItem selectedItem)
      {
         string slideName = selectedItem.Content.ToString();
         // Видалення слайда з колекції
         if (_slides.ContainsKey(slideName))
         {
            // Видалення Canvas з SlideCanvas, якщо він відображається
            if (slideCanvas.Children.Contains(_slides[slideName]))
            {
               slideCanvas.Children.Remove(_slides[slideName]);
            }
            // Видалення слайда з колекції
            _slides.Remove(slideName);
         }
         // Видалення елемента з ListBox
         SlideList.Items.Remove(selectedItem);
         // Якщо вибраний слайд був видалений, приховуємо Canvas
         if (SlideList.SelectedItem == selectedItem)
         {
            slideCanvas.Children.Clear();
```

```
                slideCanvas.Visibility = Visibility.Collapsed;
            }
            // Якщо всі слайди видалено, приховуємо Canvas
            if (_slides.Count == 0)
            {
                slideCanvas.Children.Clear();
                slideCanvas.Visibility = Visibility.Collapsed;
            }
        }
    }
}
```

ДОДАТОК Г

Лістинг файла AddStreamWindow.xaml

```
<Window x:Class="MultiStream.AddStreamWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
        mc:Ignorable="d" Title="Додати потік"
        WindowStartupLocation="CenterOwner"
        Background="#2C2C2C">
    <Grid>
        <StackPanel>
            <!-- Поле для введення назви потоку -->
            <TextBox x:Name="StreamNameTextBox"
                     Width="300"   Margin="10"
                     VerticalAlignment="Center"
                     HorizontalAlignment="Center"
                     Background="#333333"
                     Foreground="White"
                     PlaceholderText="Введіть назву потоку" />
            <!-- ComboBox для вибору типу джерела (камери або інше) -->
            <ComboBox x:Name="SourceTypeComboBox"
                      Width="300"
                      Margin="10"
                      VerticalAlignment="Center"
                      HorizontalAlignment="Center"
```

```
                Background="#333333"
                Foreground="White">
            <ComboBoxItem Content="Камера" />
            <ComboBoxItem Content="Інший потік" />
        </ComboBox>
        <!-- ComboBox для вибору камери -->
        <ComboBox x:Name="CameraComboBox"
                Width="300"
                Margin="10"
                VerticalAlignment="Center"
                HorizontalAlignment="Center"
                Background="#333333"
                Foreground="White"
                Visibility="Collapsed" />
        <!-- Кнопка для додавання потоку -->
        <Button x:Name="AddButton"
                Content="Додати потік"
                Width="300"
                Margin="10"
                VerticalAlignment="Center"
                HorizontalAlignment="Center"
                Background="#444444"
                Foreground="White"
                Click="AddButton_Click" />
    </StackPanel>
  </Grid>
</Window>
```

ДОДАТОК Д

Лістинг файла AddStreamWindow.xaml.cs

```
using AForge.Video;
using AForge.Video.DirectShow;
using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media.Imaging;
using System.Drawing;
namespace MultiStream
{
    public partial class AddStreamWindow : Window
    {
        private FilterInfoCollection videoDevices; // Список доступних камер
        public string StreamName { get; private set; }
        public string SelectedSource { get; private set; } // Назва камери

        public AddStreamWindow()
        {
            InitializeComponent();
            // Завантажуємо список камер при відкритті вікна
            LoadAvailableCameras();
        }
        private void LoadAvailableCameras()
        {
            try
            {
```

```
                videoDevices = new
FilterInfoCollection(FilterCategory.VideoInputDevice);
                CameraComboBox.Items.Clear();
                foreach (FilterInfo device in videoDevices)
                {
                    CameraComboBox.Items.Add(device.Name);
                }
                if (CameraComboBox.Items.Count > 0)
                {
                    CameraComboBox.SelectedIndex = 0; // Обрати першу камеру
за замовчуванням
                }
                else
                {
                    MessageBox.Show("Не знайдено доступних камер!",
"Помилка", MessageBoxButton.OK, MessageBoxImage.Error);
                }
            }
            catch
            {
                MessageBox.Show("Помилка при отриманні списку камер.",
"Помилка", MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }
        private void AddButton_Click(object sender, RoutedEventArgs e)
        {
            StreamName = StreamNameTextBox.Text;
```

```
            if (SourceTypeComboBox.SelectedIndex == 1) // Камера
            {
                if (CameraComboBox.SelectedItem == null)
                {
                    MessageBox.Show("Виберіть камеру!", "Помилка",
MessageBoxButton.OK, MessageBoxImage.Error);
                    return;
                }
                SelectedSource = CameraComboBox.SelectedItem.ToString();
            }
            DialogResult = true;
            Close();
        }
        private void AddStreamBTN_Click(object sender, RoutedEventArgs e)
        {
            // Якщо вибрано зображення або відео
            if (StreamList.SelectedItem != null)
            {
                var selectedStream = StreamList.SelectedItem.ToString();
                // Додамо зображення на слайд
                if (source.EndsWith(".jpg") || source.EndsWith(".png") ||
source.EndsWith(".bmp"))
                {
                    BitmapImage bitmap = new BitmapImage(new Uri(source));
                    Image image = new Image
                    {
                        Source = bitmap,
```

```
                Width = 800,
                Height = 450
            };
            // Створимо новий слайд або оновимо існуючий
            string slideName = $"Слайд {_slideCounter}";
            if (_slides.ContainsKey(slideName))
            {
                Canvas slideCanvas = _slides[slideName];
                slideCanvas.Children.Clear();  // Очищаємо попередній
вміст
                slideCanvas.Children.Add(image); // Додаємо нове
зображення
            }
            else
            {
                // Якщо слайд не існує, створюємо новий Canvas
                Canvas newSlideCanvas = new Canvas
                {
                    Width = 800,
                    Height = 450
                };
                newSlideCanvas.Children.Add(image);
                _slides[slideName] = newSlideCanvas; // Додаємо новий
слайд до колекції
            }
            // Оновлення інтерфейсу
            slideCanvas.Visibility = Visibility.Visible;
```

```
                }
                // Якщо вибрано камеру
                else if (source == SelectedSource)
                {
                    // Тут можна додати логіку для відео з камери
                    var videoCaptureDevice = new VideoCaptureDevice(source);
                    videoCaptureDevice.NewFrame += new
NewFrameEventHandler((sender, eventArgs) =>
                    {
                        Bitmap bitmap = eventArgs.Frame;
                        Image videoFrame = new Image
                        {
                            Source = BitmapToImageSource(bitmap),
                            Width = 800,
                            Height = 450
                        };
                        // Створимо новий слайд або оновимо існуючий
                        string slideName = $"Слайд {_slideCounter}";
                        if (_slides.ContainsKey(slideName))
                        {
                            Canvas slideCanvas = _slides[slideName];
                            slideCanvas.Children.Clear();  // Очищаємо попередній
вміст
                            slideCanvas.Children.Add(videoFrame); // Додаємо новий
відео кадр
                        }
                        else
```

```
                    {
                        // Якщо слайд не існує, створюємо новий Canvas
                        Canvas newSlideCanvas = new Canvas
                        {
                            Width = 800,
                            Height = 450
                        };
                        newSlideCanvas.Children.Add(videoFrame);
                        _slides[slideName] = newSlideCanvas; // Додаємо новий
слайд до колекції
                    }
                    // Оновлення інтерфейсу
                    slideCanvas.Visibility = Visibility.Visible;
                });
                videoCaptureDevice.Start();
            }
        }
        using AForge.Video.DirectShow;
        using System.Windows;
    namespace MultiStream
    {
        public partial class AddStreamWindow : Window
        {
            private FilterInfoCollection videoDevices; // Список доступних
камер
            public string StreamName { get; private set; }
            public string SelectedSource { get; private set; } // Назва джерела
```

```
        // Це список потоків, які ми будемо додавати
        public static ListBox StreamList { get; set; }
        public AddStreamWindow(ListBox streamList)
        {
            InitializeComponent();
            // Призначаємо переданий список потоків
            StreamList = streamList;
            // Завантажуємо список камер при відкритті вікна
            LoadAvailableCameras();
        }
        // Завантаження доступних камер
        private void LoadAvailableCameras()
        {
            try
            {
                videoDevices = new
FilterInfoCollection(FilterCategory.VideoInputDevice);
                CameraComboBox.Items.Clear();
                foreach (FilterInfo device in videoDevices)
                {
                    CameraComboBox.Items.Add(device.Name);
                }
                if (CameraComboBox.Items.Count > 0)
                {
                    CameraComboBox.SelectedIndex = 0; // Обрати першу
камеру за замовчуванням
                    CameraComboBox.Visibility = Visibility.Visible;
```

```
                }
                else
                {
                    CameraComboBox.Visibility = Visibility.Collapsed;
                    MessageBox.Show("Не знайдено доступних камер!",
"Помилка", MessageBoxButton.OK, MessageBoxImage.Error);
                }
            }
            catch
            {
                MessageBox.Show("Помилка при отриманні списку камер.",
"Помилка", MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }
        // Обробка натискання кнопки "Додати потік"
        private void AddButton_Click(object sender, RoutedEventArgs e)
        {
            StreamName = StreamNameTextBox.Text;
            if (string.IsNullOrEmpty(StreamName))
            {
                MessageBox.Show("Будь ласка, введіть назву потоку!",
"Помилка", MessageBoxButton.OK, MessageBoxImage.Error);
                return;
            }
            // Перевіряємо тип джерела
            if (SourceTypeComboBox.SelectedIndex == 0) // Камера
            {
```

```
                if (CameraComboBox.SelectedItem == null)
                {
                    MessageBox.Show("Виберіть камеру!", "Помилка",
MessageBoxButton.OK, MessageBoxImage.Error);
                    return;
                }
                SelectedSource = CameraComboBox.SelectedItem.ToString();
            }
            else // Інший потік
            {
                SelectedSource = "Інший потік";
            }
            // Додаємо потік до списку в головному вікні
            AddStreamToList();
            // Закриваємо вікно
            DialogResult = true;
            Close();
        }
        // Метод для додавання потоку до списку
        private void AddStreamToList()
        {
            if (StreamList != null)
            {
                StreamList.Items.Add(StreamName + " - " + SelectedSource);
            }
        }
    }  }}
```

ДОДАТОК Е
ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи:_____ Програмно-апаратні засоби для проведення інтерактивних багатоканальних відеотрансляцій_____

_____

Тип роботи:_____Магістерська кваліфікаційна робота_____

(кваліфікаційна робота, курсовий проект (робота), реферат, аналітичний огляд, інше (зазначити))

Підрозділ _____ кафедра обчислювальної техніки_____

(кафедра, факультет (інститут), навчальна група)

Керівник_____Савицька Л.А., доц. каф. ОТ_____

(прізвище, ініціали, посада)

Показники звіту подібності

Turnitin	
Оригінальність	85%
Загальна схожість	15%

Аналіз звіту подібності (відмітити потрібне)

- ✓ Запозичення, виявлені 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 автора. Роботу направити на доопрацювання.

- 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.

Заявляю, що ознайомлений (-на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається)

Автор _____ _____ Куклій Д. В._____

(підпис) (прізвище, ініціали)

Опис прийнятого рішення

_____

_____

_____

_____

_____

Особа, відповідальна за перевірку_____ _Савицька Л.А.__

(підпис)

(прізвище, ініціали)

Експерт _____

_____

(за потреби) (підпис) (прізвище, ініціали, посада)