НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

“На правах рукопису”
УДК 004.51

До захисту допущено
В. О. Завідувача кафедри
Михайло Новотарський
“__” _____ 2025 р.

Магістерська дисертація
на здобуття ступеня магістра
за освітньо-професійною програмою “Комп’ютерні системи та мережі”
зі спеціальності 123 “Комп’ютерна інженерія”
на тему: “Система дистанційного керування виводом на екран”

Виконав (-ла):
студент (-ка) ІІ курсу, групи ІО-41мп
Борисенков Роман Анатолійович _____

Керівник:
старший викладач, доктор філософії
Міщенко Людмила Дмитрівна _____

Консультант з нормоконтролю:
асистент, к.т.н.
Кулаков Олексій Юрійович _____

Рецензент:
_____

Засвідчую, що у цій магістерській
дисертації немає запозичень з праць
інших авторів без відповідних посилань
Студент (-ка) _____

Київ - 2025 року

“Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *Інформатики та обчислювальної техніки*

Кафедра *Обчислювальної техніки*

Освітньо-кваліфікаційний ступінь *магістр*

Спеціальність *123. Комп’ютерна інженерія*

Спеціалізація *123. Комп’ютерні системи та мережі*
(код і назва)

ЗАТВЕРДЖУЮ

В. О. Завідувача кафедри

Михайло Новотатарський

“__”_____ 2025 р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Борисенкову Роману Анатолійовичу

1. Тема дисертації Система дистанційного керування виводом на екран

2. Науковий керівник дисертації старший викладач, д-р.ф комп’ютерної інженерії Міщенко Л.Д. затверджені наказом по університету від “06” листопада р. № 4841-е.

3. Об’єкт дослідження Процес дистанційного керування виводом на екран та процес формування програми заходу через веб–інтерфейс.

4. Предмет дослідження методи, алгоритми та програмні рішення для розробки та впровадження системи для дистанційного керування виводом інформації на екран.

5. Перелік завдань, які потрібно розробити:

1. Провести огляд існуючих рішень.

2. Спроєктувати систему та її архітектуру.

3. Реалізувати систему для дистанційного керування виводом на екран.

4. Розробити стартап-проєкт.

6. Консультанти розділів дисертації:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Кулаков О. Ю.	03.03.2025	
2	Кулаков О. Ю.	11.05.2025	
3	Кулаков О. Ю.	01.09.2025	
4	Кулаков О. Ю.	04.10.2025	

7. Дата видачі завдання 01 вересня 2025

Календарний план

№ з/п	**Назва етапів виконання магістерської дисертації**	**Строки виконання етапів дисертації**	**Примітка**
1.	Затвердження теми роботи	01.09.25 - 07.09.25	
2.	Аналіз предметної області, огляд літератури	01.09.25 - 08.09.25	
3.	Вивчення та аналіз існуючих рішень	09.09.25 - 15.09.25	
4.	Вивчення та аналіз методів рішення поставлених задач	16.09.25 - 22.09.25	
5.	Розробка демонстраційного прототипу	23.09.25 - 13.10.25	
6.	Розробка стартап-проєкту	29.10.25 - 14.11.25	
7.	Оформлення магістерської дисертації	14.10.25 - 26.10.25	
8.	Захист	22.12.2025	

Студент _____ Роман БОРИСЕНКОВ

Науковий керівник дисертації _____ Людмила МІЩЕНКО

РЕФЕРАТ

Структура та обсяг роботи. Робота вміщає в себе вступ та чотири розділи. Загальний обсяг роботи 103 аркуші основного тексту, 15 ілюстрацій, 27 таблиць, 3 додатки. При підготовці використано літературу з 21 джерел.

Актуальність теми. Стрімка цифровізація стимулює створення рішень, які автоматизують повторювані операції людини. ПЗ, роботизовані системи та інші інструменти вже перебирають на себе повторювані завдання, звільняючи час фахівців для творчих і стратегічних задач. Запропонована робота актуальна тим, що зменшує часові витрати на повторювані процедури й частково усуває потребу в ручному контролі виведення даних на екран, підвищуючи ефективність цифрових процесів.

Мета роботи. Метою роботи є розробка програмного забезпечення, яке позбавить оператора екрану виконувати повторювані дії для підготовки медіа матеріалів, які використовуються під час заходу, в якому необхідність виводу на екран є важливою. Також, метою є впровадження системи, в якій оператор екрану зможе керувати процесом виводу на екран з будь-якої точки, не прив'язуюсь до робочого місця.

Об’єкт дослідження. Процес дистанційного керування виводом на екран та процес формування програми заходу через веб–інтерфейс.

Предмет дослідження. Методи, алгоритми та програмні рішення для розробки та впровадження системи для дистанційного керування виводом інформації на екран.

Наукова новизна. Запропоновано архітектуру розподіленої системи дистанційного управління мультимедійним контентом, яка поєднує чотири взаємопов'язані компоненти: серверний модуль на базі одноплатного комп'ютера, веб-інтерфейс для формування програми заходу, десктопну програму для відтворення контенту та мобільний додаток для дистанційного керування. Розроблено метод гібридної взаємодії компонентів системи, що базується на комбінації протоколів REST API для передачі структурованих

даних та WebSocket для синхронного керування у режимі реального часу, що забезпечує максимальну затримку відповіді 72 мс при обслуговуванні десяти одночасних підключень. Реалізовано механізм асинхронної трансляції відеопотоку екрану на мобільний пристрій оператора через протокол HTTP з MJPEG-кодуванням, що забезпечує візуальний контроль відображуваного контенту без необхідності фізичної присутності біля робочої станції. Обґрунтовано можливість побудови енергоефективної серверної інфраструктури для систем управління презентаційним контентом на базі одноплатного комп'ютера Raspberry Pi 3B з рівнем споживання електроенергії до 5 Вт, що дозволяє застосовувати систему в умовах обмежених енергетичних ресурсів та IoT-інфраструктурі.

Особистий внесок магістранта. Магістрантом самостійно розроблено систему взаємодії компонентів через мережу, а також систему захоплення другого екрану та надсилання на нього відповідного забезпечення. Також, розроблено систему живого редагування програми.

Практична цінність. Оперативний обмін даними між різними обчислювальними пристроями, забезпечення електронною обчислювальною машиною послідовність виведення інформації й демонстрування мобільному пристрою результат виконання отриманих команд.

Ключові слова. ОПЕРАТОР ЕКРАНУ, СЕРВЕР, ЕКРАН, ПРОГРАМА ЗАХОДУ, ВЕБ СОКЕТИ, МЕРЕЖЕВИЙ ПРОТОКОЛ, ВИВІД, ОПТИМІЗАЦІЯ, ВІДДАЛЕНЕ КЕРУВАННЯ.

ABSTRACT

Structure and scope of work. The work contains an introduction and four chapters. The total volume of the work is 103 pages of the main text, 15 illustrations, 27 tables, 3 appendices. Literature from 21 sources was used in the preparation.

Relevance of the topic. Rapid digitalization stimulates the creation of solutions that automate repetitive human operations. Software, robotic systems and other tools are already taking over repetitive tasks, freeing up specialists' time for creative and strategic tasks. The proposed work is relevant in that it reduces the time spent on repetitive procedures and partially eliminates the need for manual control of data output to the screen, increasing the efficiency of digital processes.

Purpose of the work. The purpose of the work is to develop software that will relieve the screen operator from performing repetitive actions to prepare media materials used during an event in which the need for output to the screen is important. Also, the goal is to implement a system in which the screen operator will be able to control the process of output to the screen from any point, not tied to the workplace.

Object of research. The process of remote control of output to the screen and the process of forming the event program via a web interface.

Object of research. The process of remote control of the display on the screen and the process of forming the event program via a web interface.

Subject of research. Methods, algorithms and software solutions for developing and implementing a system for remote control of information displayed on the screen.

Scientific novelty. The architecture of a distributed system for remote control of multimedia content is proposed, which combines four interconnected components: a server module based on a single-board computer, a web interface for forming the event program, a desktop program for playing content and a mobile application for remote control. A method of hybrid interaction of system components has been developed, based on a combination of REST API protocols for transferring structured data and WebSocket for synchronous control in real time, which provides a maximum response delay of 72 ms when servicing ten simultaneous connections. A mechanism

for asynchronously broadcasting a screen video stream to the operator's mobile device via the HTTP protocol with MJPEG encoding has been implemented, which provides visual control of the displayed content without the need for physical presence at the workstation. The possibility of building an energy-efficient server infrastructure for presentation content management systems based on a Raspberry Pi 3B single-board computer with a power consumption level of up to 5 W has been substantiated, which allows the system to be used in conditions of limited energy resources and IoT infrastructure.

Personal contribution of the master's student. The master's student independently developed a system for interacting components via the network, as well as a system for capturing a second screen and sending appropriate software to it. Also, a system for live program editing has been developed.

Practical value. Operational data exchange between different computing devices, ensuring the electronic computer's sequence of information output and demonstrating to the mobile device the result of executing the received commands.

Keywords. SCREEN OPERATOR, SERVER, SCREEN, EVENT PROGRAM, WEB SOCKETS, NETWORK PROTOCOL, OUTPUT, OPTIMIZATION, REMOTE CONTROL.

ЗМІСТ

ВСТУП

У сучасному світі людський інтелект та час є цінними ресурсами, особливої уваги заслуговує проблема автоматизації повторюваних процесів. Виконання циклічних операцій, які можуть бути спрощені технічними системами, призводить до нераціонального використання інтелектуального потенціалу фахівців та зниження ефективності робочих процесів.

Серед проблемних напрямків є організація та проведення конференцій, семінарів та інших публічних заходів, де є необхідність у виведенні мультимедійного контенту на екран. Традиційний підхід до управління медіаконтентом характеризується низкою недоліків, серед яких надсилання матеріалів через соціальні мережі в останні хвилини перед початком заходу, що створює додаткову стресову ситуацію та навантаження на оператора екрану. Процес перенесення та впорядкування контенту в програмному забезпечення вимагає часові витрати та може призводити до стресових ситуацій. Крім того, необхідність роботи з декількома вікнами для перегляду послідовності програми та матеріалів, які необхідно відображати, ускладнює процес управління.

Існує технічна проблема. пов’язана з використанням презентаторів спікерами, які під час виступів використовують власні пристрої для перегляду конспекту та презентатори для перемикання слайдів. Однак радіус дії таких пристроїв, зазвичай, є обмеженим, через що сигнал не завжди досягає обчислювального пристрою з презентацією. Це призводить до необхідності голосових команд в сторону оператора екрану для перемикання, що порушує хід презентації та знижує професійність заходу.

Мета роботи є розробити таку систему, яка забезпечить автоматизацію управління мультимедійним контентом на публічних заходах, мінімізує необхідність ручного втручання оператора та дасть можливість керувати матеріалом дистанційно.

Розроблена система передбачає перенесення процесу підготовки програми заходу та завантаження матеріалів у сховище через веб-інтерфейс. Організатори матимуть доступ до формування структури програми, завантаженню матеріалів різних форматів: тексти, зображення, слайди, відео та аудіо. Вся інформація зберігається на сервері, що усуває необхідність прямої комунікації з оператором екрану на етапі підготовки.

Функціональність системи дозволить оператору підключатися до запущеного на робочій станції програмного забезпечення через смартфон або планшет. Після синхронізації програма заходу буде завантажуватись на мобільний пристрій, що дозволить здійснювати управління відображенням з будь-якої точки конференц-залу.

Для спікерів система пропонує рішення, яке об’єднує функцію перегляду та перемикання слайдів, а також перегляд конспекту до промови. Це усуває проблему обмеженого радіусу дії традиційних презентаторів та забезпечує безперебійне керування презентацією через інтернет–з’єднання.

РОЗДІЛ 1
ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Огляд альтернативного програмного забезпечення

На сучасному ринку програмного забезпечення представлено низку продуктів для управління виводом мультимедійного контенту на екран. Проте, аналіз існуючих рішень виявив відсутність систем, яків повній мірі забезпечують функціональність дистанційного керування у локальних мережевих середовищах.

Запропонована система характеризується складною системою, що передбачає розгортання серверної частини, веб-інтерфейсу, десктопної програми та мобільний клієнт. Така архітектура орієнтована на застосування в обмежених мережевих середовищах, що потребує первинного налаштування на місці проведення заходу, що обмежує можливості масового розповсюдження без технічного супроводу.

Водночас, архітектуру системи можливо спростити шляхом скорочення веб-інтерфейсу та серверного компонента. Цей підхід передбачає автономне функціонування десктопної програми з вбудованою функціональністю формування програми заходу та прикріплення медіаматеріалів. Встановлене з’єднання з мобільним пристроєм можна здійснити використовуючи QR-код, або ж рядок підключення, який генерується в налаштуваннях десктопної програми. Після синхронізації оператор екрану отримує можливість здійснювати керування виводом використовуючи бездротовий зв’язок. Саме з позиції спрощеної архітектури здійснюється порівняльний аналіз існуючих програмних рішень, представленихи на ринку.

ProPresenter

Це програмне забезпечення є професійним комерційним рішенням від компанії Renewed Vision. Створювалось для живих заходів, богослужінь,

конференцій та трансляцій, що стало можливим завдяки потужній графічній та відео підсистемі, здатна мати багато виходів, сценічних дисплеїв, інтеграцію з Бібліями, піснями та можливістю потокового виводу [1].

Основними її можливостями є крос-платформенність, має можливість виводити одночасно на декілька екранів, є можливість виводити фон, як в якості фото, так в якості і відео. Це рішення є професійним, тому воно і коштує досить дорого - 1197 грн в місяць за одну людину, або ж за кампус до 20 людей - 2435 грн в місяць. Також, за якісний вивід потокового відео це програмне забезпечення відповідно вимагає і достатньо потужне залізо. Його мінімальними вимогами є [2] :

1. Операційна система: macOS 12.7.4 або новіша, або Windows 10 v1903 або новіша;
2. Процесор: Intel i5 (або еквівалентний AMD) або Apple Silicon (M1 або новіший);
3. Оперативна пам’ять: 8GB;
4. Графічна карта: виділена відеокарта з щонайменше 2GB VRAM для виводу в HD якості.

Для виводу ж в якості 4K та на декілька екранів є такі вимоги:

1. Процесор: Intel i7 або i9 (або еквівалентний AMD) або ж Apple Silicon;
2. Оперативна пам’ять: 16GB;
3. Графічна карта: виділена відеокарта з 8GB+ VRAM та оновлені драйвера.

Загальний вигляд головного вікна програми можна побачити на Рис. 1.1.

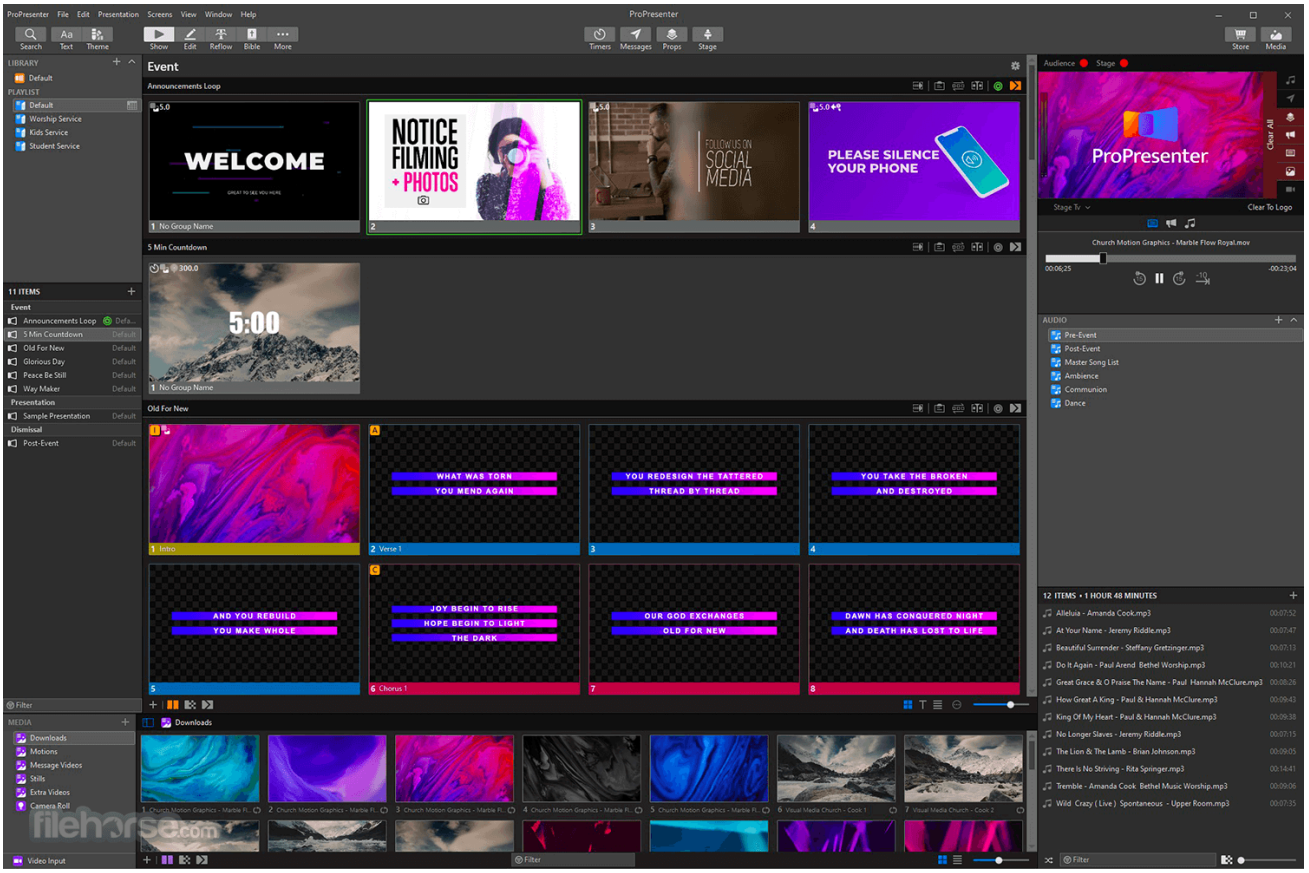

Рис. 1.1 - головне вікно програми ProPresenter

Як ми можемо побачити з зображення вище, інтерфейс цієї програми ділиться на декілька зон, в кожної з яких є своє призначення. Розглянемо кожну з них.

1. Бібліотека (зліва зверху): ця зона відповідає за збереження в собі повного списку пісень та презентацій. Наприклад, можна створити одну бібліотеку, в якій будуть розміщені тільки пісні українською. а в іншій тільки англійською мовами;
2. Плейлист (зліва посередині): цей блок, по суті, відображає список програм, або ж певну послідовність пісень та презентацій. В нього можна додавати нові блоки, змінювати їх порядок, видаляти;
3. Медіа бібліотека (знизу): відображає в собі такі матеріали як зображення або відео. Якщо виводити їх на екран, вони будуть відображатись на фоні пісень;

4. Аудіо бібліотека (справа знизу): відповідає за запуск звукових доріжок, відповідно можна створювати різні плейлисти, в кожному з яких буде відображатись список аудіо;
5. Передпоказ (справа зверху): відповідає за керування відео та аудіо, а також відображає копію того, що відбувається на сусідньому екрані;
6. Панель налаштувань (зверху): відповідає за різні режими, в тому числі режим показу, режим редагування, теми, повідомлення, таймери тощо.

Загалом, якщо оцінювати цю програму, то вона є досить сучасною і оптимізованою для роботи з відео, текстом, слайдами та іншими медіа компонентами.

Недоліки застосунку

Якщо казати про недоліки застосунку, то основним є досить висока ціна, яка буває непідйомною для невеликих організацій. Також, саме головне вікно програми є досить зручним у взаємодії, тим не менш, там немає можливості застосовувати drag and drop технологію, також не достатньо зручно може бути додавання різних форматів медіа, наприклад для того, щоб додати слайдшоу з архіву зображень, прийдеться кожне зображення перетягувати на нову сторінку слайду і відцентровувати його, що не є зручно.

OpenLP

OpenLP є безкоштовним рішенням з відкритим кодом, в основному він орієнтований на богослужіння в церквах. Він добре підходить для показу пісень, біблійних віршів, зображень, створення презентацій та плейлистів. Встановити його можна на Windows, macOS та Linux [3].

Основними можливостями цього програмного забезпечення є те, що воно пітримує багато платформ, в ньому є основні функції показу, в ньому є можливість попереднього перегляду та підтримка декількох дисплеїв. Також є можливість дистанційного керування через веб-інтерфейс чи мобільний пристрій. Цей застосунок має активну спільноту, розвиток та стабільну документацію.

Мінімальними вимогами до цього програмного забезпечення є [4]:

1. Операційна система: Windows 10/11, Linux, macOS 10.8 або новіша;
2. Процесор: 2 Ghz;
3. Оперативна пам’ять: 2 GiB;
4. Дисплей: підтримка декількох моніторів.

Загальний вигляд головного вікна програми можна побачити на Рисунку 1.2

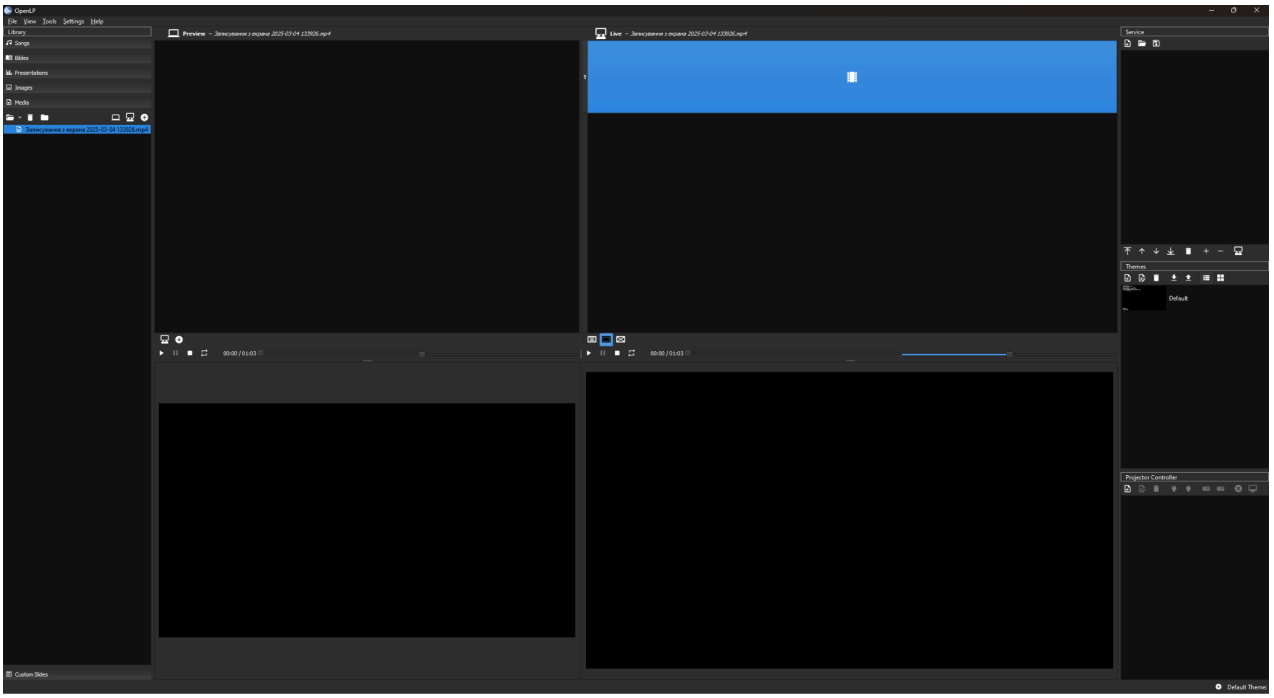

Рис .1.2 - головне вікно програми OpenLP

Вікно цієї програми також розподіляється на декілька вікон, суть яких полягає в наступному:

1. Медіа бібліотека (зліва): панель, в якій можна додавати медіа, тексти пісень та презентації;
2. Менеджер програми (зправа): відповідає за формування програми в послідовності їх виконання, такі як пісні, біблійні тексти, відео, зображення;
3. Панель передпоказу: відображає поточний елемент програми. Тут користувач може переглянути як буде виглядати контент на екрані до того, як він його виведе;
4. Живий вихід: окреме вікно, яке транслюється на проектор.

Недоліки застосунку

Основною проблемою цього застосунку є незручний дизайн. Наприклад, панель бібліотеки зліва є не зручною в плані перемикання вкладок з різними типами контенту. Також, немає зручного drag and drop, замість цього потрібно використовувати кнопки, з якими потрібно ще ознайомитись.

EasyWorship

Це програмне забезпечення є комерційним рішенням, орієнтоване на церкви, свою увагу акцентує на простоту та швидкість підготовки програми. [5]

Основними можливостями цієї програми є інтуїтивно зрозумілий інтерфейс, якому можна досить швидко навчитись працювати. Є можливість створення повної програми заходу в одному місці. Окрім цього, є можливість інтеграції з NDI, має можливість використовувати відео, зовнішні виходи, прозорі фони та теми. Є програма для дистанційного керування під iOS та Android, що дозволяє керувати виводом з телефону або планшета. Мінусом є те, що ця програма працює лише на Windows. Також, це програмне забезпечення не є безкоштовним, що може бути критично для малих церков та організацій.

Мінімальними вимогами щодо характеристик комп’ютера є [6]:

1. Операційна система: Windows 10/11;
2. Процессор: 2.4 GHz Intel i3 (або аналоги AMD);
3. Оперативна пам’ять: 4 GB;
4. Графічна карта: NVIDIA GeForce GT x30.

Також, для стабільної роботи рекомендується така конфігурація:

1. Операційна система: Windows 10/11 64 bit;
2. Процессор: 2.5 - 3.4 GHz Intel I5 або i7 на 4 ядра (або аналоги AMD);
3. Оперативна пам’ять: 8 GB;
4. Графічна карта: NVIDIA GeForce GTX/RTX.

Загальний вигляд головного вікна програми можна побачити на Рисунку 1.3

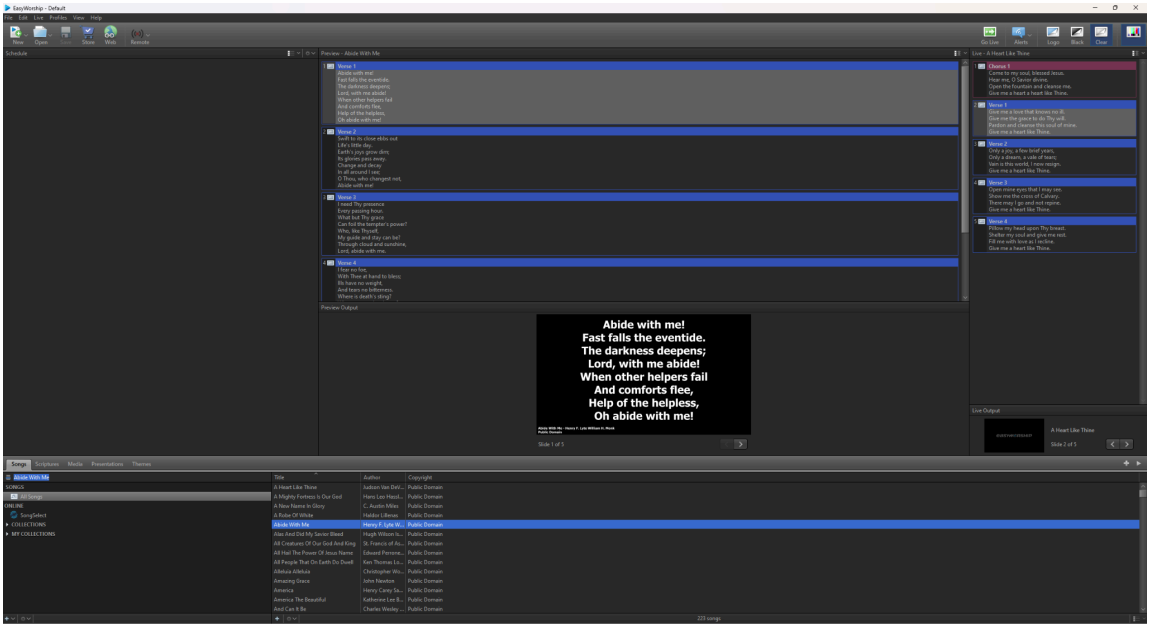

Рис. 1.3 - головне вікно програми EasyWorship

Як можна побачити з цього рисунку, в цій програмі є декілька панелей, у кожної з якої є своя задача. Верхня центральна відповідає за передпоказ слайдів чи куплетів текстів. Панель нижче відображає те, що програма виводить зараз на другий екран. Нижня панель містить в собі бібліотеку з пісень, медіа, тем та Писання.

Недоліки застосунку

Не дивлячись на те, що програма вимагає досить незначних ресурсів для виконання поставлених задач, у порівнянні з іншими конкурентами, головним недоліком цього ПЗ можна назвати його досить незручний та застарілий дизайн. Незручність полягає в тому, що панелі розташовані дещо в хаотичному порядку, немає ніякої послідовності. Головною задачею дизайну бути інтуїтивно зручним, чого не можна сказати про цю програму.

FreeShow

Є безкоштовною платформою, створена спеціально для церков, щоб надати багато функцій без витрат на комерційний продукт, має відкритий код, підтримує такі функції як медіа, відео, шаблони та віддалене керування. [7]

Основними можливостями є:

1. Програма є безкоштовною і не вимагає ліцензії, відповідно немає витрат за програмне забезпечення;
2. Має досить широкий спектр функцій: редактор слайдів, є можливість редагування тексту на льоту, шаблони, фон, відео, зображення;
3. Має Multiple Views - різні виходи для сцени, аудиторії, стріму, є NDI підтримка, таймери та зворотні відліки;
4. Має вбудоване віддалене керування: через телефон або веб-інтерфейс;
5. Має сумісність з такими форматами як PowerPoint, ProPresenter, EasyWorship, OpenSong та інше.

Мінімальними вимогами для цього ПЗ є [8]:

5. Операційна система: Windows 10/11;
6. Процессор: 2.9 GHz Intel i5 9400 (або аналоги AMD);
7. Оперативна пам’ять: 8 GB;
8. Графічна карта: NVIDIA GeForce GT x30.

Загальний вигляд головного вікна можна побачити на рисунку 1.4.

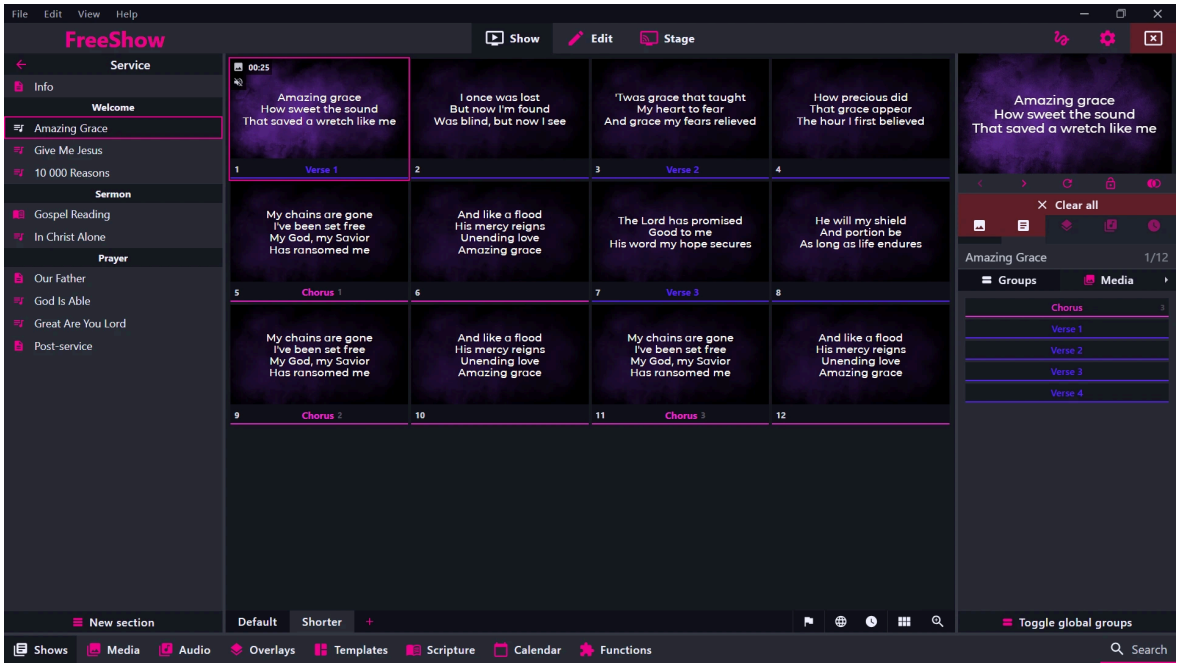

Рис. 1.4 - головне вікно програми FreeShow

Зверху можна побачити головне вікно програми FreeShow. На ньому можна побачити по суті ті ж самі панелі, що були і в інших порівняльних застосунках. Серед таких є:

1. Знизу бібліотека з різними типами медіа контенту;
2. Вікно з програмою заходу;
3. Передпоказ слайдів, натискаючи на які посилається сигнал на вивід;
4. Вікно стану, яке відображає те, що відображає на головному екрані.

Недоліки програми

Серед особливостей програми, які важко назвати недоліками, можна назвати лише те, що її можна запустити на операційній системі windows. Також, що більш реально назвати недоліком, вона вимагає досить великих ресурсів, як з точки зору процесора, так і з точки зору відео пам’яті.

1.2 Порівняння існуючих рішень

В цьому підрозділі буде здійснено порівняльний аналіз існуючих таблиць. Це необхідно зробити для того, щоб наяву можна було переглянути всі переваги та недоліки того чи іншого застосунку, поєднати їх у одне та створити таку систему, в якій не було б місця для цих недоліків. Отже, на таблиці 1.1 можна побачити порівняльну таблицю всіх вищезгаданих програм для виводу на екран.

Таблиця 1.1

Порівняльна таблиця існуючих рішень

Критерій	Моя система	ProPresenter	OpenLP	EasyWorship	FreeShow
Ціна / ліцензування	Безкоштовна	Комерційна	Безкоштовна	Платна	Безкоштовна

Продовження таблиці 1.1

Критерій	Моя система	ProPresenter	OpenLP	EasyWorship	FreeShow
Платформа	Linux, macOS	Windows, macOS	Linux, macOS, Windows	Windows	Windows, macOS
Мультидисплей	Ні	Так	Так	Так	Обмежено
Медіапідтримка (відео, фон, зображення)	Частково	Так	Так	Так	Базово
Зручність інтерфейсу	Так	Професійний	Простий	Інтуїтивний	Так
Біблія / пісні /CCLI інтеграція	Частково	Так	Так	Так	Так
Потік / NDI / стріминг	Так	Так	Частково (через плагіни)	Так	Залежить від версії
Підтримка	–	Так	Активна спільнота	Так	Активна спільнота

Продовження таблиці 1.1

Критерій	Моя система	ProPresenter	OpenLP	EasyWorship	FreeShow
Керування з телефону	Так	Так	Так	Ні	Через локальний веб-сервер
Формування програми через веб–інтерфейс	Так	Ні	Ні	Ні	Ні
Можливість on-premise розгортання	Так	Так	Так	Так	Так

Загалом, якщо виходити з результатів порівняльної таблиці, то можна сказати, що сьогодні ринок пропонує як професійні комерційні рішення, так і повноцінні безкоштовні альтернативи. Можна сказати, що між ними є одне головне розділення - функціональність та рівень інтуївнивної зрозумілості. Наприклад, такі професійні інструменти як ProPresenter або EasyWorship орієнтовані на роботу у великих середовищах, наприклад в церквах, конференційних залах або студіях. Вони мають такий набір можливостей як: багатодисплейна підтримка, вивід потокового відео, NDI, графічні шари та синхронізація декількох екранів. Такі рішення надають стабільність та зручність, але вимагають ліцензії й глибокого розуміння системи та того як вона працює.

З іншої точки зору, є безкоштовні аналоги, такі як OpenLP або Freeshow, які орієнтовані на простоту, відкритість і гнучкість. Вони не мають такого гарного інтерфейсу або розкішних візуальних ефектів, тим не менш їх цілком достатньо для того, щоб виконувати основні завдання, як от відображення пісень, біблійних текстів, зображень або коротких відео. Особливою привабою є можливість керування презентацією з телефона через веб-інтерфейс.

З точки зору стабільності та підтримки, платні продукти звичайно мають офіційні канали підтримки та документацію, тоді як безкоштовні спираються на активну спільноту. Водночас відкриті проєкти часто розвиваються завдяки внескам ентузіастів і пропонують міжплатформені можливості, що робить їх більш універсальними. В таблиці 1.2 відображено мінімальні вимоги до апаратного забезпечення для того, щоб запустити те чи інше ПЗ.

Таблиця 1.2

Порівняльна таблиця мінімальних вимог до комп’ютера

Компонент	ProPresenter	OpenLP	EasyWorship	Freeshow
Операційна система	macOS 17.7.4, Windows 10	Windows 10, Linux, macOS	Windows 10	Windows 10
Процесор	Intel i5, Apple Silicon M1	2 ГГц	2.4 ГГц Intel i3	2.9 ГГц intel i5 9400
Оперативна пам’ять	8 Гб	2 ГіБ	4 Гб	8 Гб
Відеокарта	2 Гб VRAM	Підтримка декількох моніторів	NVIDIA GeForce GT x30	NVIDIA GeForce GT x30

Оглядаючи таблицю зверху можна побачити, що найменш вимогливою є програма OpenLP. Вона підтримує всі основні операційні системи та вимагає надзвичайно мало потужностей.

1.3 Актуальність та доцільність розробки

Щоденно у світі проводяться тисячі заходів, які потребують професійного мультимедійного супроводу. Сюди входять як освітні заклади, такі як університети, так і масштабні концертні програми. Кожен з цих заходів вимагає відповідного технічного забезпечення для відображення медіаматеріалів. Рівень складності програмного забезпечення варіюється в залежності від масштабу заходу: для шкільних уроків, зазвичай, достатньо використання Microsoft PowerPoint, для проведення ж заходів в актових залах чи конференц-центрах необхідне спеціалізоване програмне забезпечення для управління керуванням медіа.

Традиційно, підхід організацій характеризується необхідністю в операторі екрану на фіксованому робочому місці, що обмежує гнучкість переміщення. Актуальність даного дослідження полягає в розробці системи, яка мінімізує необхідність прив’язки оператора до конкретного робочого місця та автоматизує процеси підготовки програми.

Запропонований підхід передбачає перенесення функцій формування програми заходу на організаторів через веб-інтерфейс, що усуває необхідність передачі матеріалів через соціальні мережі та ручної обробки оператором. У перспективі така система може трансформувати роль технічного оператора до такого рівня, що необхідність в його присутності буде мінімальною, оскільки основна відповідальність за медіа супровід буде лежати на організаторах програми. Його задачею буде лише забезпечення стабільності мережевого зв’язку та вирішення інших технічних питань, у разі якщо такі будуть виявлені.

Доцільність розробки такої системи зумовлена низкою факторів.

Економічна доцільність. Комерційні програмні рішення професійного рівня характеризуються високою вартістю ліцензування та підтримки, що робить їх недоступними для установ з обмеженим бюджетом. Вони, зазвичай, вимагає достатньо потужне апаратне забезпечення, яке теж може бути недоступним для деяких груп, у яких є проблеми з фінансами.

Технологічна доцільність. Існуючі рішення часто характеризуються надмірною складністю інтерфейсу та вимагають спеціальних компетенцій. Альтернативні рішення з відкритим кодом можуть поступатися у зручності інтерфейсу та ергономіці. Запропонована система реалізує принцип мінімалізму, який передбачає собою спрощення інтерфейсу та автоматизацію підготовчих процесів, що знижує поріг входження для користувачів без спеціальної технічної підготовки.

Енергетична доцільність. Існуюче програмне забезпечення в міру використання потужного апаратного забезпечення, відповідно, вимагає і потужного енергетичного живлення для комп’ютерів, які використовуються. Поточне ж рішення дозволить значно скоротити енерговитрати шляхом використання в якості сервера одноплатного комп’ютера Raspberry Pi 3B. Цей пристрій з обмеженими обчислювальними можливостями може забезпечувати роботу сервера на рівні 5 Вт. Це означає, що система зможе мати живлення навіть від звичайного повербанка від 4 годин в залежності від його об’єму.

Організаційна доцільність. Традиційна схема комунікації між організаторами та операторами екрану здійснюється через соціальні мережі, що призводить до втрати часу та інтелектуальних ресурсів на обробку матеріалів, особливо у випадку надходження контенту перед початком заходу. Система з централізованим веб-інтерфейсом стандартизує процес підготовки та усуває необхідність ручної обробки оператором.

1.4 Постановка задачі

Задачею дослідження є розробка програмної системи дистанційного управління виводом мультимедійного контенту на проекційне обладнання, яке забезпечує автоматизацію процесів підготовки програми заходу та мінімізує залученість оператора екрану на робочому місці. Для досягнення поставленої задачі необхідно вирішити наступні задачі:

1. Розробити архітектуру розподіленої системи, яка дозволить підключатись одночасно десятьом клієнтам до програми керування медіаконтентом заходу;
2. Забезпечити споживання електроенергії серверною частиною на рівне не вище 5V;
3. Розробити веб–інтерфейс, в якому буде можливість формувати програму заходу;
4. Включити в архітектуру підтримку базових форматів файлів, таких як текст, jpeg, png та pptx;
5. Розробити ергономічний інтерфейс для користувачів, що забезпечить інтуїтивно зрозуміле розташування панелей та компонентів;
6. Провести функціональне тестування розробленої системи та верифікувати досягнення поставлених цілей.

Результатом виконання роботи має стати функціонуюча програмна система, яка задовольняє всі вищезазначені вимоги та забезпечує ефективне вирішення проблем.

ВИСНОВКИ ДО РОЗДІЛУ 1

У першому розділі проведено комплексний аналіз існуючих програмних рішень для управління презентаційним контентом та виявлено їх основні недоліки, що обґрунтовує актуальність розробки нової системи дистанційного керування виводом на екран.

Проведений порівняльний аналіз виявив ключові недоліки існуючих систем: необхідність ручного формування програми заходу оператором, прив'язаність до стаціонарного робочого місця, обмежений радіус дії бездротових презентаторів (що призводить до втрати сигналу), відсутність можливості дистанційного формування програми через веб-інтерфейс, високе споживання електроенергії (від 30 до 300 Вт).

Обґрунтовано актуальність та доцільність розробки з трьох позицій: економічної, технологічної, організаційної та енергетичної.

Сформульовано чітку постановку задачі з шістьма конкретними підзадачами: розробка розподіленої архітектури для 10 одночасних підключень, забезпечення споживання не вище 5 Вт, створення веб-інтерфейсу для формування програми, підтримка базових форматів файлів, розробка ергономічного інтерфейсу та проведення функціонального тестування.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА СИСТЕМИ

2.1 Проєктування функціоналу

Проєктування функціоналу програми є важливою частиною розробки програмного продукту. Насамперед, це важливо з тієї точки зору, що щоб чогось досягти, потрібно розуміти, куди ти йдеш, або окреслити межі того, що ти вважаєш завершеним завданням. Саме цьому завданню і буде посвячений цей розділ.

Першим кроком в проєктуванні системи буде створення UML-діаграми варіантів. Ця діаграма є важливою в контексті розробки програмного забезпечення, оскільки в ній розробники спочатку візуалізують взаємозв’язки між акторами (користувачами або іншими системами) та варіантами використання (функції або послуги), що надаються системою. Вона використовується для моделювання функціональних вимог та бізнес-процесів системи, показуючи, хто і як взаємодіє з системою. Завдяки цьому, діаграма надає спільну мову для обговорення функціональних потреб системи між розробниками, аналітиками та замовниками, що зменшує непорозуміння. Також, вона допомагає ілюструвати можливі помилки в структурі системи на ранніх етапах розробки, що надає можливість виправити їх якомога швидше.

2.2 Опис використовуваних технологій

Якщо розглянути всі технології, які використовуються в роботі, то можна побачити, що їх можна розділити на два типи - фреймворки, за допомогою яких створюється додатки та протоколи, які використовуються всередині цих додатків. Відповідно до цих типів і буде здійснено опис.

2.2.1 Опис фреймворків

Angular

Це фреймфорк для розробки динамічних односторінкових веб–інтерфейсів [9]. Він побудований на TypeScript і надає інструменти для роботи з даними в прямому ефірі. Основними характеристиками цієї платформи є:

1. Побудова веб–інтерфейсу на основі компонентів: вся структура складається з незалежних компонентів, кожен з яких відповідає за певну частину інтерфейсу;
2. Жива взаємодія між бекендом та фронтендом: фреймворк забезпечує двостороннє зв’язування даних, завдяки чому зміни даних відразу відображаються в інтерфейсі;
3. Впровадження залежностей: механізм Dependency Injection забезпечує керування об’єктами та сервісами, і тим самим зменшує кількість повторного коду та підвищуючи гнучкість архітектури;
4. Маршрутизація: дозволяє створювати багатосторінкову навігацію та переходити між компонентами, передавати параметри в URL і керувати станом сторінок.

Qt C++

Це фреймворк для створення переважно комп’ютерних програм із графічним інтерфейсом. Для цього він може використовувати як C++, так і Python для бекенду, а також QML для фронтенду [10]. Його основними характеристиками є:

1. Кросплатформність: є можливість використовувати, по суті, один і той самий код під такі платформи як Windows, Linux та macOS;
2. Архітектура складається з модулів, таких як: models, providers, controllers, core та інше;

3. Робота через сигнали: для взаємодії cpp коду з qml використовуються так звані сигнали для того, щоб передати певну інформацію або виконати візуальну зміну;
4. Підтримує архітектуру Model-View-Controller.

Java

Це мова програмування, яка дозволяє розробляти додатки в тому числі і для ОС Android [11]. Головними особливостями побудови програми під Android на цій мові є:

1. Життєвий цикл Activity: кожен екран проходить через фази, такі як onCreate(), onStart(), onResume(), onPause, onStop, onDestroy();
2. Розмітка інтерфейсу на XML: інтерфейс зазвичай описується у XML-файлах, які знаходяться у папці res/layout;
3. Чітка структура Android-проєкту: дозволяє зручно орієнтуватися в проєкті, має декілька основних директорій для збереження коду, тестування та налагодження;
4. Робота з мережею: дозволяє працювати з такими протоколами як WebSocket, RESTful API та HTTP;
5. Є офіційна IDE Android Studio для розробки проєкту.

2.2.2 Опис використовуваних протоколів

WebSocket

Це протокол зв’язку, який забезпечує двостороннє з’єднання між клієнтом та сервером. Його головною особливістю у порівнянні з іншими протоколами є те, що він не вимагає щоразу відкривати з’єднання, коли відправляється запит. Відповідно до цього, як тільки клієнт та сервер встановлюють один між одним з’єднання, вони можуть обмінюватись даними в реальному часі. За рахунок одного постійного з’єднання також здійснюється менше навантаження на

мережу. Цей протокол є ідеальним для застосунків, які вимагають швидкого обміну даними [12].

В нашому випадку, цей протокол буде використовуватись для живої передачі блоків програми, які можуть змінюватись в реальному часі, а також для передачі медіа контенту через мережу, завдяки можливості відправляти їх в бінарній послідовності.

RESTful API

Це протокол для взаємодії між клієнтом та сервером через HTTP-запити, де дані передаються у форматі JSON або XML. На відміну від WebSocket, цей протокол не є таким реактивним, і зазвичай використовується для обміну великими масивами даних, наприклад списками вибірок з бази даних [13].

В нашому випадку саме так він і буде використовуватись - для передачі таких списків як пісні, інформація про слайди, слайдшоу та програми заходів.

HTTP з MJPEG–шифруванням

Ця технологія дозволяє поширювати відео, закодоване як Motion JPEG через протокол HTTP. Його особливістю є простота та широка сумісність, починаючи з браузерів, і закінчуючи програмами, в які ця технологія вбудована. MJPEG шифрує кожний відео кадр окремо як зображення, що спрощує декодування. Використання HTTP забезпечує легке інтегрування та високу суміність з різними платформами та пристроями і є ефективним навіть в мережах зі строгими конфігураціями. Є чудовим рішення для відео в реальному часі, дозволяє налаштовувати розширення кадрів [14].

В випадку трансляції стану екрану на смартфон важливий рівень затримки, оскільки з програми, яка виводить матеріал на екран буде захоплюватись матеріал та повинен фактично в прямому ефірі відображати його на смартфон, щоб оператор міг орієнтуватись про поточний стан екрану.

SQLite 3

Це легка кросплатформна вбудована база даних, яка зберігає всі дані у одному файлі, що є її головною перевагою. Сама собою воно представляє

невелику, швидку, надійну, багатофункціональну, SQL двигун. Є найбільш використовуваним двигуном бази даних у світі [15]. Її основними характеристиками є:

1. Працює локально: все зберігається безпосередньо на пристрої користувача у файлі. Це дозволяє працювати з базою без підключення до мережі;
2. Підтримує SQL-запити: повністю сумісна з більшістю стандартів SQL, тому з нею можна працювати через команди SQL.
3. Не потребує підключення через клієнт-сервер: не має серверної частини. Програма напряму звертається до файлу, що спрощує розгортання;
4. Швидка: завдяки відсутності викликів через мережу забезпечує високу швидкість читання за запису, особливо якщо обсяги невеликі;
5. Її можна легко вбудувати в будь-які мобільні телефони.

Postgresql

Це об’єктно-реляційна система керування базами даних [16]. Основними її перевагами є можливість підключення до неї через мережу, а також можливість зберігання найрізноманітніших форматів даних у своїх таблицях. Особливо привабливою є можливість зберігання навіть масивів або json тексту. Її основними характеристиками є:

1. Підключення через адресу та порт: працює як сервер, до якого можна підключитись. Доступ можна обмежувати у файлі pg_hba.conf, встановлюючи дозволені адреси та інше;
2. підтримує SQL-запити: повністю сумісна з стандартами SQL, крім основних запитів можна працювати з транзакціями, join-ами, підзапитами, агрегатами, тригерами та процедурами;
3. Зберігання різних типів даних: має широкий набір типів, таких як: числові, текстові, дата та час, масиви, json, uuid, геометричні типи, xml, бінарні дані, мережеві типи, власні типи користувача;
4. Розширена система ролей та прав доступу: права можуть задаватись на бази даних, схеми, таблиці, стовпці, функції;

5. Вбудований пошук: індекси GIN а GiST, формування ts_vector / ts_query, морфологічний аналіз, кастомні словники, врахування стоп-слів, ранжування результатів;
6. Надійність та відмовостійкість: використовує WAL, що гарантує цілісність даних навіть при раптовому вимкненні сервера. Також, підтримує реплікацію, резервні копії, відновлення після збою, можливість будування кластерів.

2.3 Проєктування архітектури системи

Наступним кроком буде проєктування системи. Це важливо з тієї точки зору, що це допоможе з самого початку уникнути непотрібних помилок та неспівпадінь між класами, назвами полів та їх типами. В цьому розділі, для проєктування схем баз даних та архітектури використовується платформа draw io [17].

В цій системі основними форматами виведення буде текст пісні та слайдшоу. Отже, необхідно зобразити схему того, як будуть виглядати таблиці в базі даних та які зв’язки між ними будуть. Почнемо з того, як буде зберігатися пісня в базі даних. На рисунку 2.1 можна побачити схему таблиць пісня та збірник.

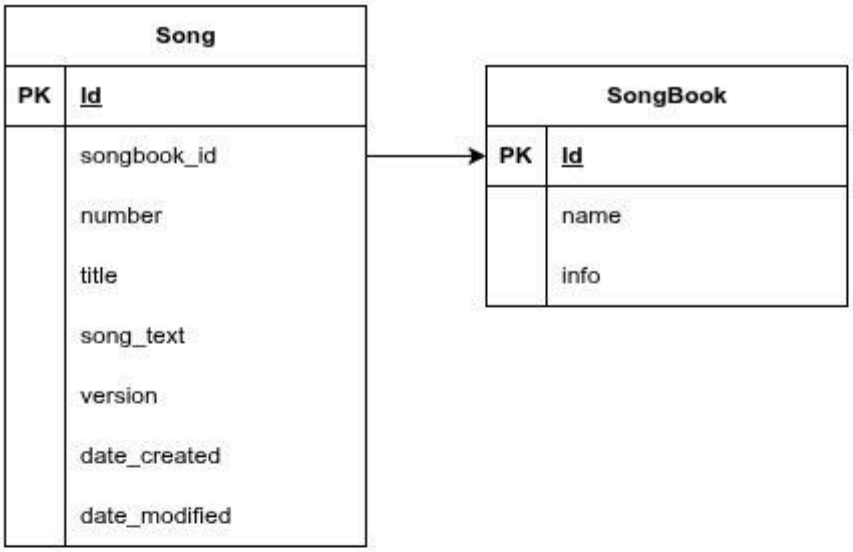

Рис 2.1 - Схема таблиць Song та SongBook та їх пов'язаність в базі даних

Як можна побачити з рисунку, в системі буде міститись дві таблиці. Суть першої таблиці полягає в тому, що вона містить в собі поля для зберігання основних параметрів пісні, такі як назва, текст, версія, дата створення, дата останньої модифікації. Але також там є і вказівник на таблицю збірник, яка містить в собі лише id, назву та інформацію про збірник. Задача цієї таблиці полягає в тому, щоб в різних альбомах можна було зберігати пісні і при пошуку конкретної пісні можна було б застосувати відповідні фільтри, щоб швидше знайти те, що потрібно.

Наступним кроком є проєктування схеми для слайд шоу. Суть цієї системи взаємопов'язаних таблиць полягає в тому, що в одному слайдшоу може зберігатись одночасно декілька зображень (слайдів), відповідно потрібно розробити таку схему реляційної бази даних, щоб це можна було зберегти. На рисунку 2.2 можна побачити схему того, як пов'язані між собою реляційні таблиці слайдшоу та слайдів.

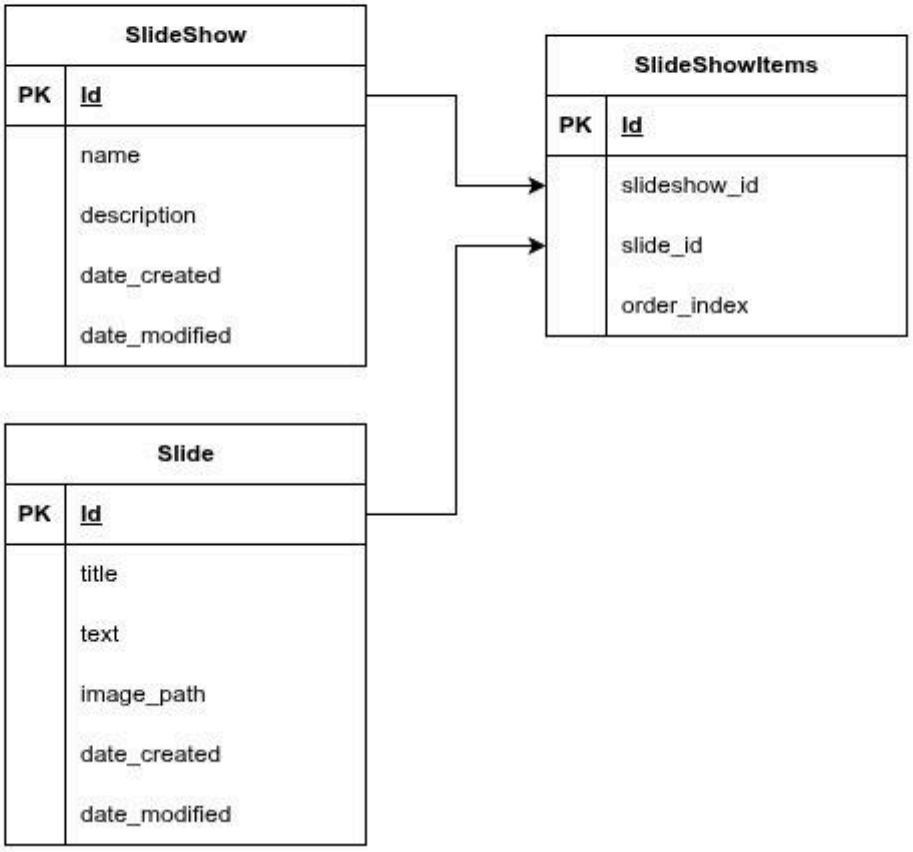

Рис. 2.2 - схема таблиць SlideShow, Slide, SlideShowItems та зв'язки між ними

Як можна побачити зі схеми вище, для того, щоб зберігати інформацію про слайдшоу, потрібно мати відразу 3 таблиці. Суть таблиці SlideShow полягає в тому, що вона містить просто ід, назву презентації, опис та дату створення та останньої модифікації. В наступній таблиці Slide вже зберігається конкретна інформація про кожний окремий слайд. Сюди входить ід слайду, його назва, текст - це може бути коментар до конкретного слайду, шлях до файлу, дату створення та останньої модифікації. І нарешті третя таблиця, суть якої полягає в тому, щоб поєднати дві попередні. Вона містить в собі: власне ід, ід слайдшоу, ід слайду та порядковий номер слайда в слайдшоу. Відповідно, для того, щоб зберегти одне слайдшоу, наприклад, з 5 зображеннями, в цій таблиці буде 5 записів, де в першому полі буде вказано ід слайдшоу, в другому ід слайду і в третьому полі - який порядок він буде займати.

,Вище описані два основні типи даних, які можна виводити на екран. Але ця система має дещо більше даних для зберігання, тому далі будуть спроектовані ще деякі типи даних, які є необхідними для функціонування системи.

Як вже згадувалося в поставлених задачах, ціль цієї роботи полягає в тому, щоб спростити взаємодію між оператором екрану та виконавцями тих чи інших блоків програми, яким потрібно щось вивести на екран. Отже для того, щоб це спростити, потрібно спроектувати таблицю, яка буде зберігати в собі одну список програм, а також потрібно сформувати вигляд файлу, який буде зберігати в собі окремі блоки (елементи) програми.

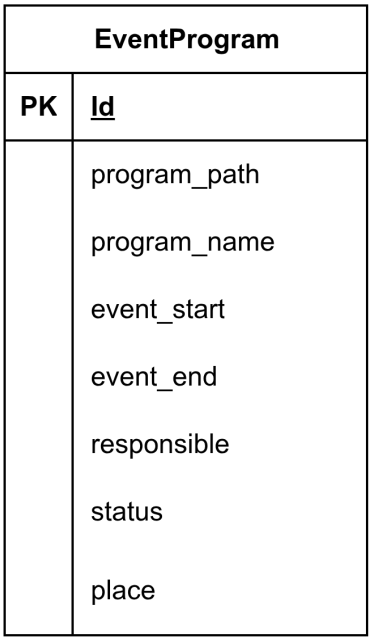

Рис. 2.3 - схема таблиці EventProgram, яка зберігає дані про програми

Вище можна побачити спроєктовану таблицю, яка відображає сутність кожної окремої програми. Тут міститься такі поля як: шлях до файлу, в якому зберігаються блоки програми, назва програми, дата початку та закінчення заходу, відповідальна людина за організацію зустрічі, статус заходу та місце, де цей захід буде відбуватись.

На рисунку 2.4 відображені поля для зберігання інформації про кожний блок програми.

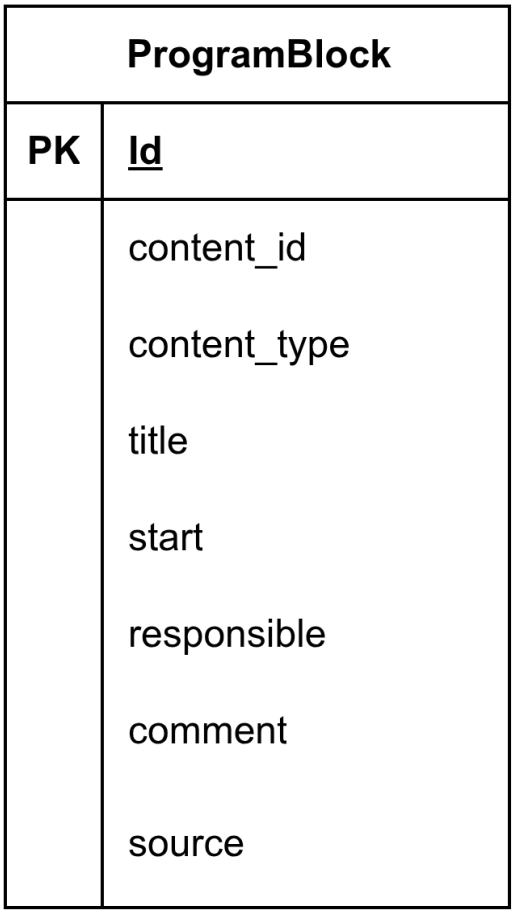

Рис. 2.4 - схема блоку програми

В рисунку вище можна побачити такі важливі поля як: ід елементу, ід контенту, тип контенту (пісня або слайдшоу), назва блоку, час початку, відповідальна людина за цей блок, коментар за необхідності та перетворений медіа контент для цього поля.

Якщо казати про те, як виглядає структура самого файлу, то це може виглядати наступним чином:

{
“program” : {
“id”: int,
“name” : String,
“path” : String

},

“blocks” : [

{

“id” : int,

“type” : String,

“title”: String,

“start” : String,

“responsible” : String,

“comment” : String,

“source”: String

}

],

“updated_at” : DateTime,

“version” : int

}

З опису цього файлу можна побачити, що він містить в собі такі поля як ід, ім’я та шлях. Це зроблено для того, щоб можна було зручно шукати записи в базі даних з файлом.

Наступним кроком вважаю за необхідне створити use-case діаграму системи для того, щоб внести в ясність те, як вона може виглядати. Відповідна схема відображена на рис 2.5.

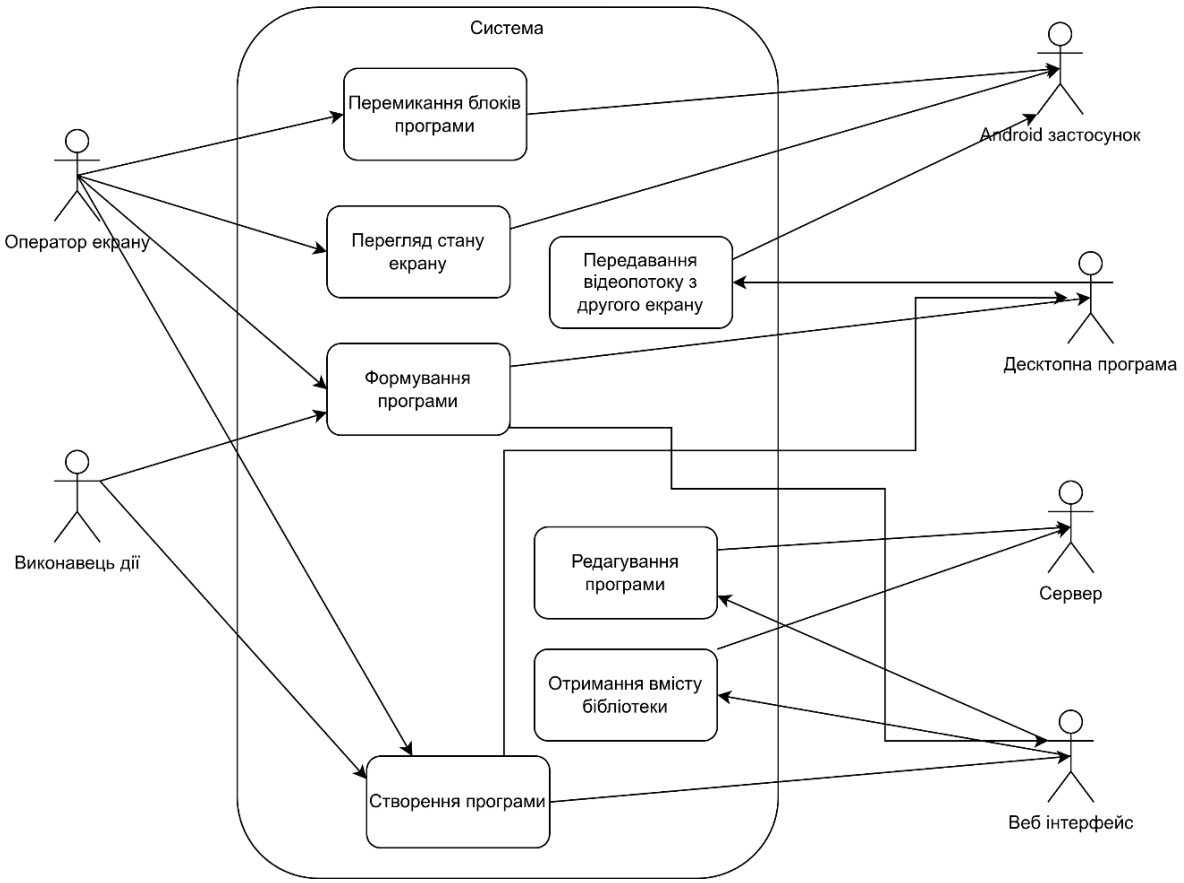

Рис. 2.5 - use case діаграма системи

Як можна побачити зі схеми, тут не так багато функцій, які виконує система. Звичайно, якщо розглядати цю систему в більшому масштабі, то функцій там дещо більше, тут же зображені основні категорії дій.

2.4 Загальна інфраструктура проєкту

Тепер, після того, як були описані основні технології для створення системи, можна перейти до безпосередньої розробки системи. В ході цього не буде демонструватись весь написаний код, оскільки його на стільки багато, що його вистачило б на ще 3 такі роботи, особливо з огляду на те, що іноді один файл може містити в собі до 1000 рядків коду. Тут буде продемонстрована основна структура проєкту, а також те, як воно все разом взаємодіє.

Якщо казати про загальний вигляд проєкту, то, як вже згадувалося раніше, він складається з декількох програм. Якщо казати про те, як загалом виглядає ця інфраструктура та як вона пов’язана одна з одним, то це можна побачити на рисунку 2.6.

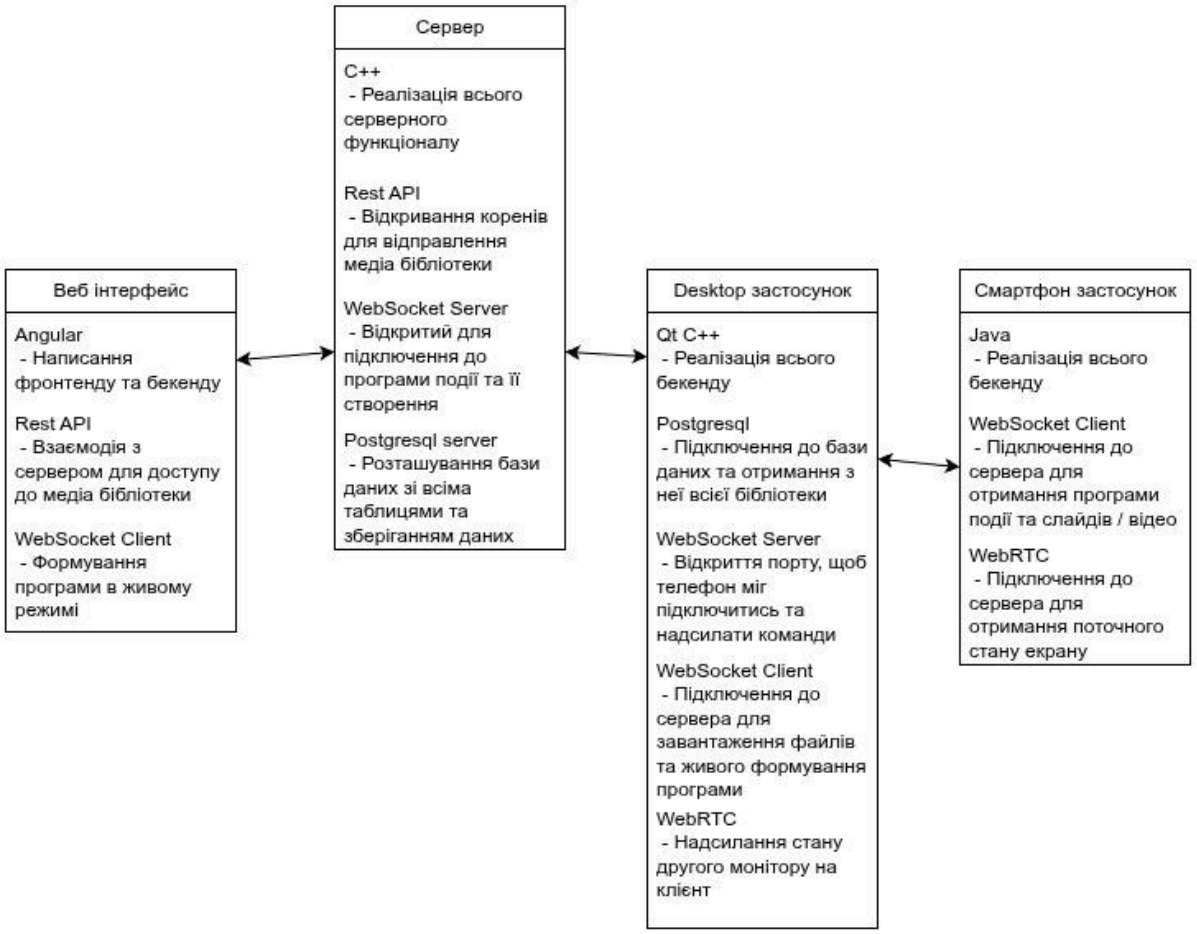

Рис. 2.6 - схема зв’язків компонентів системи та технології, що в них використовуються

На цьому рисунку можна побачити різні проєкти та основні фреймворки, які в них використовуються внутрішньо або для взаємодії з іншими компонентами однієї системи. Розглянемо кожен елемент системи детальніше.

2.5 Сервер

Сервер є дуже важливою частиною системи, оскільки він забезпечує зберігання та синхронізацію певних даних між клієнтами. В контексті цього проєкту, його головною ціллю є збереження даних про програму та її вміст (блоки), також однією з його ролей є зберігання таких даних як текст та слайдшоу. Для того, щоб це стало можливим, сервер використовує декілька мережевих протоколів, в кожного з яких є певна окрема задача, а також бібліотеку для зберігання даних. Тут буде розглянуто те, як відповідні протоколи використовуються для роботи сервера.

RESTful API

В контексті цієї роботи, протокол RESTful API використовується для таких цілей як:

1. Створення, отримання, редагування та видалення елементів зі списку програм;
2. Створення, отримання, редагування та видалення елементів зі списку пісень та списків збірників;
3. Створення, отримання, редагування та видалення елементів з списку слайдшоу та слайдів.

Для виконання саме цих задач цей протокол використовується не випадково. В цьому випадку немає необхідності отримувати певний масив даних в режимі реального часу. Відповідно, цей список можна передати одним великим пакетом, який і відобразиться у веб-інтерфейсі, або ж в десктопній програмі.

Якщо казати про дані, які сервер отримує та передає, то тут можна спостерігати наступну картину:

1. /api/songs:
    a. GET - отримання списку пісень;
    b. GET /search - пошук пісень за назвою або за номером;
    c. POST - створення пісні;

   d. PUT - оновлення пісні за ід;
   e. DELETE - видалення пісні за id;

2. /api/songbooks:
   a. GET - отримання списку збірників;
   b. POST - створення нового збірника;
   c. PUT - оновлення створеного збірника за ід;
   d. DELETE - видалення збірника за ід;
3. /api/slideshows:
   a. GET - отримання повного списку слайдшоу;
   b. GET /search - отримання списку слайдшоу за ключовим словом;
   c. POST - створення нового слайдшоу;
   d. PUT - оновлення створеного слайдшоу за ід;
   e. DELETE - видалення слайдшоу за ід;
4. /api/slides:
   a. GET /search - пошук слайдів за назвою;
5. /api/programs:
   a. GET - отримання списку програм;
   b. GET /search - пошук програм на назвою;
   c. POST - створення нової програми;
   d. PUT - оновлення програми за ід;
   e. DELETE - видалення програми за ід.

Таким чином клієнт може отримувати ті дані, які йому потрібні, звертаючись за необхідним йому шляхом та вказуючи йому потрібні параметри.

Якщо казати про відповідь сервера, то він у відповідь кладе два поля, завдяки чому відповідь має такий формат:

{
“success” : Bool,
“data” : []
}

Це допомагає клієнту зрозуміти чи отримав він відповідь успішно, чи ні.

WebSocket

Цей протокол дозволяє передавати дані в режимі реального часу по байтам. В цьому контексті, він використовується для наступних задач:

1. Зберігання програми в реальному часі;
2. Отримання слайдів та відео в живому режимі;

В цьому контексті, сервер також очікує, що до нього дані будуть приходити в певному вигляді для того, щоб він міг їх правильно трактувати та надіслати коректну відповідь. В залежності від того, що хоче отримати клієнт, сервер очікує відповідні формати повідомлень. Для того, щоб зрозуміти що відправляти, сервер очікує в повідомленні поле “action”, в якому можуть бути такі варіанти та, відповідно додаткові поля, які вони очікують:

1. “program/open” - відкрити програму;
2. “program/set” - встановити сесію поточної програми;
3. “program/save” - примусове збереження програми;
4. “program/close” - закрити програму.

Для того, щоб сервер розумів, про яку програму йде мова, до нього можуть передаватись наступні параметри: “program_path” - шлях до файлу програми, “programId” - ід програми.

Також, для взаємодії клієнта з сервером щодо отримання та надсилання слайдів, виділені інші додаткові поля “action”, щоб це стало можливим. Ось деякі з них:

1. “upload_image” - викликається при початку завантаження зображення на сервер, очікує таку структуру даних: {“filename”: String, “totalSize” : long int, “mimeType”: String};
2. “upload_image_chunk” - надсилається при завантаженні частини зображення. По суті, через цю дію відправляються метадані до кожної частинки зображення в такому форматі: {“chunkIndex”: int, “isLastChunk” : Boolean, “totalSent” : int};

3. “get_slide_image” - призначається для відправлення лише одного слайду, отримуючи від клієнта такий запит: {“slideId” : int}.

4. “get_image” - призначається для відправлення на клієнт конкретне зображення слайду. По суті, сервер отримує ід слайду та відправляє на клієнт по частинах зображення. Відповідь він надсилає в такому вигляді: {“action”: String, “imageid” : int, “fileName” : String, “fileSize” : int, “mimeType”: String}. Разом з цими мета даними відправляє саме зображення на клієнт по частинах.

SQLite

В ході створення інфраструктури для системи також вирішено, що серверна частина буде використовувати базу даних на основі SQLite. головними даними, які вона буде зберігати, будуть:

1. тексти / пісні;
2. Інформація про слайди та слайдшоу;
3. Інформація про заходи.

Відповідно, щоб це стало можливим була спроєктована база даних SQLite. Таблиці з її схемами були зображені в розділі 2.3.

Побудова проєкту

Оскільки програма сервера розроблена на мові програмування C++ варто згадати про те, як побудувати та запустити її. Попередньо, для того щоб це зробити, потрібно встановити бібліотеки, які використовуються в ході розробки. В таблиці 2.1 можна побачити пакети, які потрібно встановити для того, щоб проєкт міг запуститися, а також яка за що відповідає.

Таблиця 2.1

Необхідні бібліотеки для збірки програми сервера

Назва пакету	Призначення
build-essential, cmake	інструменти збірки

Продовження таблиці 2.1

Назва пакету	Призначення
qt6-base-dev, qt6-base-dev-tools	Qt6 Core та базові компоненти
qt6-websockets-dev, qt6-httpserver-dev	Робота з мережевими протоколами
libboost-filesystem-dev, libboost-log-dev, libboost-system-dev	робота з файлами, логами та кодами помилок
libsqlite3-dev, libpq-dev, postgresql-client	робота з БД, SQLite 3 та Postgresql
sudo apt update && sudo apt install build-essential cmake qt6-base-dev qt6-base-dev-tools qt6-websockets-dev qt6-httpserver-dev libboost-log-dev libboost-filesystem-dev libboost-system-dev libsqlite3-dev libpq-dev postgresql-client	

Після встановлення потрібних пакетів здійснюється формування файлу для побудови проєкту, задача якого включити в себе всі потрібні бібліотеки, всі потрібні файли .cpp та .h та, використовуючи CMake, скомпілювати проєкт у виконуваний файл і запустити його. Лістинг такого файлу CMakeLists.txt можна знайти в Додатку Б.

Для того, щоб побудувати відповідний проєкт, потрібно виконати наступні кроки [18]:

1. створити папку build та перейти в неї: *mkdir build && cd build*;
2. Запустити генерацію файлів збірки, шукаючи файл CMakeLists.txt у батьківській директорії: *cmake ..*;
3. Запустити компіляцію проєкту: *make*.

Після виконання цих кроків у директорії *build/* можна буде знайти виконуваний файл ServerApp, який можна запустити або через провідник, або в терміналі: *./build/ServerApp*.

Особливості побудови проєкту під raspberry OS

Для того, щоб побудувати проєкт під цю платформу, бажано використовувати достатньо комп'ютер з достатньою потужністю, чого не можна сказати про саму raspberry pi [19]. Проблема полягає в тому, що цей міні комп'ютер використовує процесор arm через що можуть бути проблеми, пов'язані з виконанням вихідного файлу. Натомість є альтернативне рішення. Ось кроки для здійснення цього рішення:

1. Встановити на Linux ARM компілятор: *sudo apt install gcc-aarch64-linux-gnu g++-aarch64-linux-gnu;*
2. Створити папку: *mkdir ~/rpi-sysroot;*
3. скопіювати з raspberry pi папки: *scp -r pi@IP:/lib ~/rpi-sysroot/ && scp -r pi@IP:/usr ~/rpi-sysroot/;*
4. створити файл: *nano ~/rpi-toolchain.cmake;*
5. Вставити туди код з Додатку А;
6. Запустити CMake: *mkdir build-rpi && cd build-rpi && cmake .. -DCMAKE_TOOLCHAIN_FILE=~/rpi-toolchain.cmake;*
7. Зкомпілювати: *make -j$(nproc);*
8. відправити скомпільований файл на raspberry: *scp ServerApp pi@IP:/home/pi.*

Після виконання відповідних кроків програму можна буде запустити на міні комп'ютері raspberry pi виконавши: */home/pi/ServerApp*.

2.6 Веб–інтерфейс

Складова цієї частини системи є необхідним елементом системи, але його відсутність не завадить виводити зображення на екран. Тим не менш, тут будуть

коротко описані технології, які використовуються для реалізації взаємодії з іншою частиною системи.

Задачею веб-інтерфейсу є можливість створення програми дистанційно. Це необхідно для економії часу та ресурсів, які спрямовані на виконання повторюваної роботи. Відповідно, щоб це стало можливим, розроблено дві основні сторінки, задача яких полягає в наступному: перша відображає список програм та певні дані про них, в другій можна редагувати конкретну програму, додаючи до в ній блоки, перетягуючи туди медіа елементи.

Відповідно до цього запиту були розроблені графічні елементи, зображені на рис. 2.7.

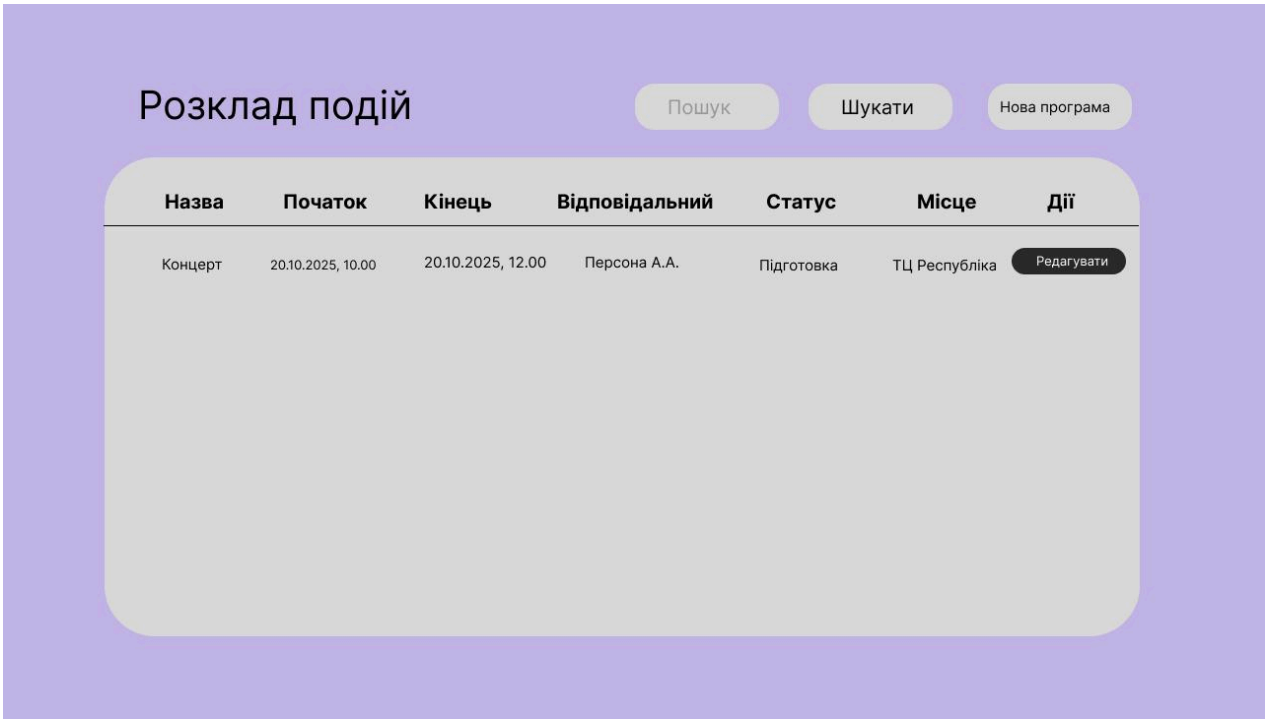

Рис. 2.7 - приблизне візуальне відображення головної сторінки з програмами

На цьому зображенні можна побачити вигляд головної сторінки. Як можна побачити, тут буде відображатись список програм, буде доступний пошук, а також можливість створення нової програми. Нижче, на рис. 2.8 зображено сторінку для редагування програми.

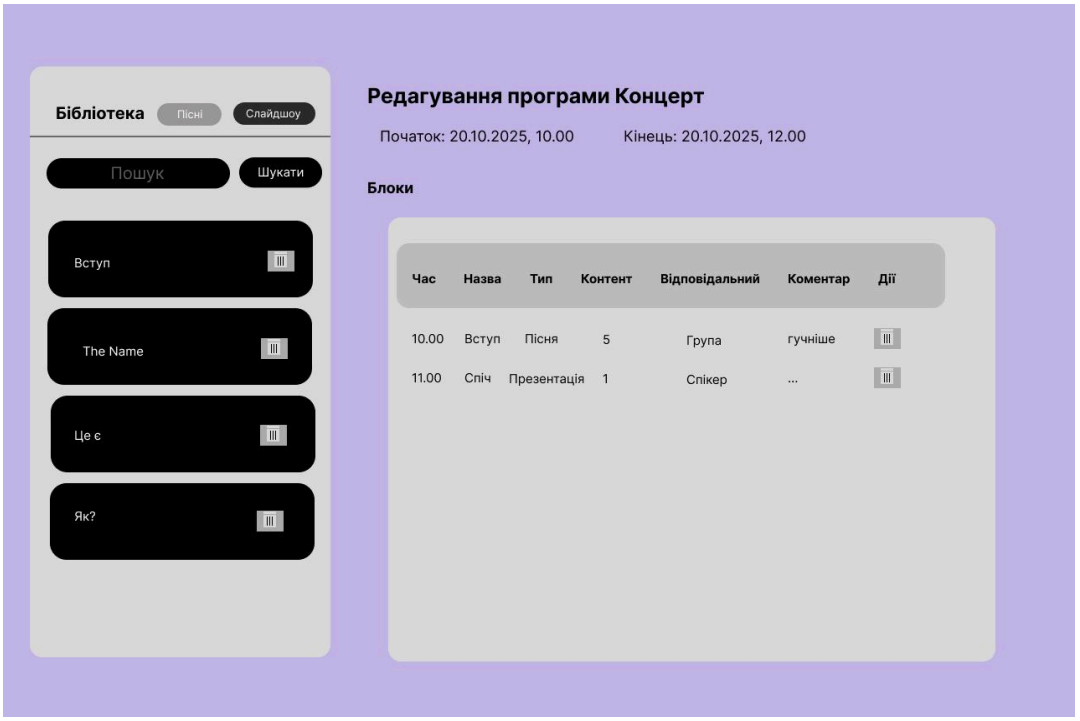

Рис. 2.8 - відображення сторінки редагування програми

На цьому рисунку можна побачити такі компоненти, як бібліотека, інформація про програму, а також безпосередньо самі блоки, які можна перетягувати в потрібній послідовності. Також, є можливість перетягування пісень або слайдшоу прямо на блок програми, щоб воно прив'язалось до певного блоку.

Варто відзначити, що на цій сторінці кожний компонент описаний в окремому файлі. Це дозволяє слідувати чіткій структурі, яку зручно редагувати.

Побудова проєкту

Для того, щоб побудувати проєкт на Angular для початку потрібно встановити відповідний фреймворк та бібліотеки [20]. Для цього потрібно в терміналі виконати наступні команди:

1. Встановлення node.js: *curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/instgall.sh | bash && nvm install node && nvm use node;*
2. Встановлення Angular CLI: *sudo npm install -g @angular/cli;*
3. Перевірити, що все успішно встановлено: *ng version.*

Оскільки суть самого проєкту є запуск веб–інтерфейсу, то в даному випадку для запуску програми не потрібно виконувати збірку проєкту в один файл. Щоб запустити веб–інтерфейс на рівні тестування в localhost достатньо у директорії проєкту виконати наступну команду: *npm start.* Вона дозволяє запустити веб–інтерфейс за посиланням http://*localhost:4200*, при переході на яке в браузері можна буде побачити його.

2.7 Десктопна програма

Ця частина системи є однією з найважливіших, адже завдяки ній виконуються такі основні функції, як безпосередній вивід на екран, а також приймання сигналів на вивід. Саме ця програма має найвищий сенс в своєму існуванні, навіть якби не були реалізовані всі інші частини системи.

Якщо казати про бібліотеки, яка вона в собі використовує, то тут їх найбільше: Qt, два WebSocket, RESTful API та SQLite3. В міру того, що в попередніх програмних продуктах вже розповідалось про роботу WebSocket, то тут варто лише вказати на те, що тут вони використовуються по суті для тих самих цілей, що і у веб–інтерфейсі та сервері. Те ж саме можна сказати і про RESTful API, задача якого полягає лише в тому, щоб отримати список програм з сервера. Те ж саме можна сказати і про SQLite3. Його головною задачею є збереження таких записів як пісні, слайдшоу, збірники та програми локально.

Особливу увагу в цьому розділі можна звернути саме на будову проєкту, а також візуальне відображення десктопної програми. Нижче описана будова проєкту, а також яку роль відіграє кожен з компонентів окремо.

1. */resources* - папка з файлами дизайну qml
    a. */icons* - збереження всіх іконок програми;
    b. */components* - зберігає компоненти, які відображають певну сутність;
    c. */dialogs* - зберігає компоненти діалогових вікон;

d. */display* - зберігає компонент, який захоплює другий екран і виводить на нього надіслану інформацію;
e. */panels* - зберігає в собі панелі бібліотеки, програми, передпоказу, виводу;
f. */tabs* - зберігає в собі вкладки пісень та слайдшоу, що відображаються у бібліотеці;
g. /*utilities* - зберігає утиліти для стилів та qml логування;
h. main.qml - дизайн головної сторінки, який збирає в собі всі компоненти до купи;

2. /src - директорія, що містить в собі файли коду .cpp та .h.

a. /*controllers* - зберігає в собі контролер, який відправляє сигнали на вивід;
b. /*cors* - зберігає в собі такі базові елементи, які відповідають за взаємодію з БД, WebSocket, RESTful API, ініціалізація компонентів та інше; */models* - зберігає в собі моделі даних, як от пісні, слайдшоу, програма та інше.
c. */providers* - містить в собі компоненти, які відповідають за взаємодію файлів дизайну qml з моделями даних, наприклад для взаємодії з піснями - SongProvider.
d. *main.cpp* - початкова точка, яка ініціалізує програму, підвантажує всі бібліотеки та запускає головний файл зовнішнього вигляду main.qml;
e. *resources.qrc* - зберігає в собі посилання на всі компоненти, що розташовані в папці resources.

Якщо казати про те, як все це разом виглядає, то це можна побачити на рисунку 2.9.

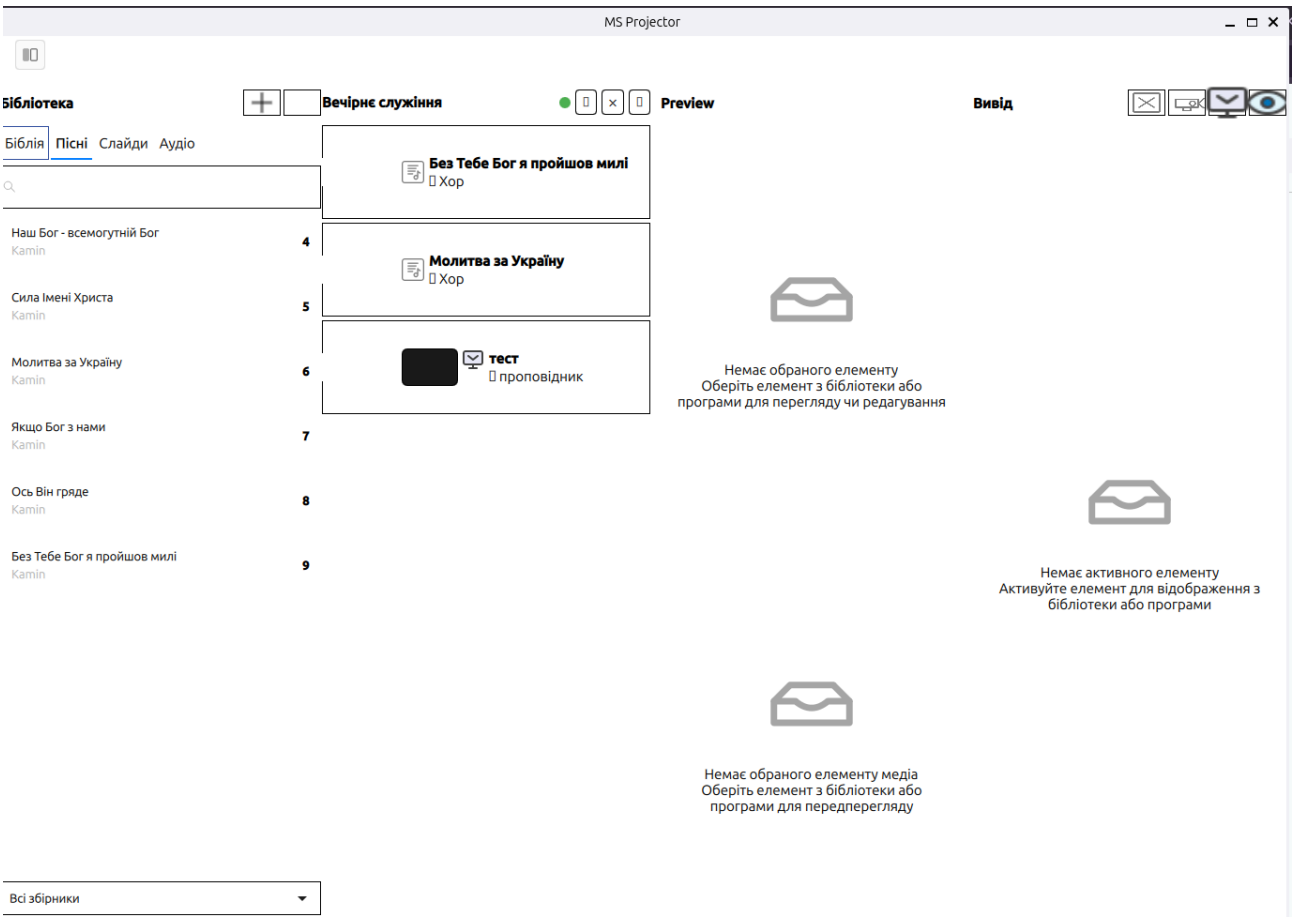

Рис. 2.9 - головне вікно десктопної програми

Як можна побачити з рисунку вище, програма розділена на декілька 4 панелі, у кожної з яких є своя задача чи то роботи з бібліотекою, програми, передпоказу чи виводу. Особливістю цього рішення є те, що всі ці компоненти знаходяться в певній не випадковій послідовності. Це зроблено з тих міркувань, щоб програма була дійсно інтуїтивно зрозумілою. Однією з цілей такого інтерфейсу є мінімальна кількість кнопок. Тепер перенесення різних елементів з панелі в панель здійснюється за допомогою технології drag and drop, суть якої полягає в тому, що ти можеш перетягувати елементи, зажавши їх мишкою.

Також, є задум мати можливість створити програму таким чином, щоб в ній можна було виводити декілька елементів на один екран. Наприклад, щоб на фон можна було виводити стилізоване зображення, а поверх нього якийсь текст.

Побудова проєкту

В міру того, що десктопна програма розроблена на C++ варто відзначити особливості побудови відповідного проєкту. Програма розроблялася в середовищі Cursor. Ця платформа була вибрана, оскільки вона має доступ до мовних моделей, які допомагали у пришвидшенні виконання роботи. По-суті, ця програма є абсолютним аналогом Visual Studio Code лише з тією відмінністю, що вона має розширений доступ до штучного інтелекту.

Для того, щоб побудувати проєкт, потрібно встановити ряд пакетів. В таблиці 2.2 можна побачити список потрібних пакетів та команду для їх встановлення.

Таблиця 2.2

Необхідні бібліотеки для збірки десктопної програми

Назва пакету	Призначення
build-essential, cmake	інструменти збірки
qt6-base-dev, qt-base-dev-tools	Qt6 Core та базові компоненти
qt6-declarative-dev, qt6-quick-dev	Робота з QML та динамічними інтерфейсами
qt6-quickcontrols2-dev	Надання елементів інтерфейсу
qt6-widgets-dev	Робота з віджетами
qt6-sql-dev	Робота з SQL запитами
qt6-multimedia-dev	Робота з медіа форматами
qt6-svg-dev	читання SVG формату
qt6-websockets-dev	Робота з Веб сокетами

Продовження таблиці 2.2

Назва пакету	Призначення
libpq-dev, postgresql-client	Робота з базами даних
qt6-base-dev-bin, qt6-base-private-dev, qt6-declarative-private-dev, qt6-webengine-private-dev	Runtime бібліотеки для запуску
sudo apt update && sudo apt install -y build-essential cmake qt6-base-dev qt6-base-dev-tools qt6-declarative-dev qt6-quick-dev qt6-quickcontrols2-dev qt6-widgets-dev qt6-sql-dev qt6-multimedia-dev qt6-webengine-dev qt6-svg-dev qt6-websockets-dev libpq-dev postgresql-client	

Після встановлення потрібних пакетів здійснюється формування файлу для побудови проєкту, задача якого включити в себе всі потрібні бібліотеки, всі потрібні файли .cpp та .h та, використовуючи CMake, скомпілювати проєкт у виконуваний файл і запустити його. Лістинг такого файлу CMakeLists.txt можна знайти в Додатку В. Приклад того, як побудувати проєкт можна знайти в розділі 2.4.1 аналогічно побудові серверної програми.

2.8 Android програма

Ця частина системи займає друге місце у важливості реалізації поставленої задачі. Саме вона забезпечує можливість виводу на екран певний контент дистанційно.

Android програма, на відміну від всіх інших частин системи, не буде настільки великою. Її функціонал не надто широкий, що не заважає йому бути корисним. Головними функціями, якими він буде наділений будуть:

1. отримати відкриту в десктопному застосунку програму;

2. відправляти команду та контент на вивід;
3. отримувати поточний стан екрану.

Виходячи з цих поставлених задач було спроєктовано візуальний вигляд того, як буде виглядати застосунок, який зображено на рисунку 2.10.

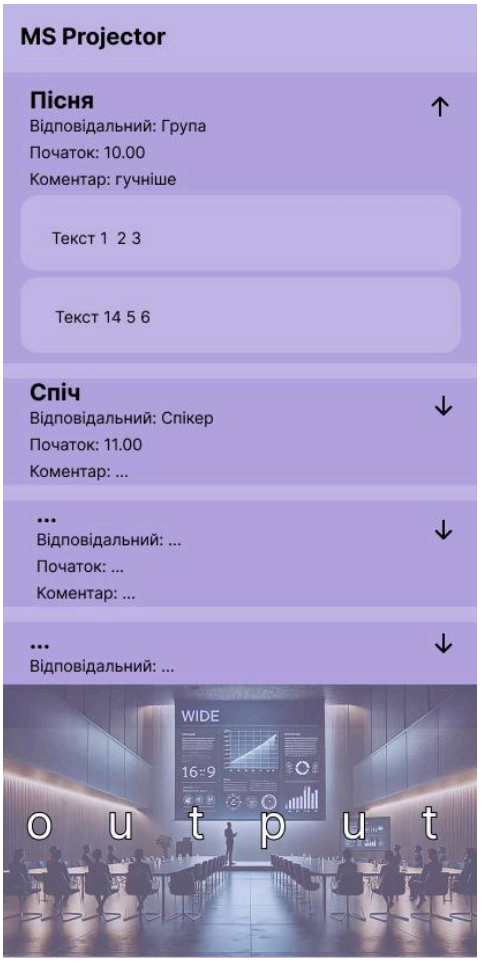

Рис 2.10 - головне вікно Android застосунку

На рисунку вище можна побачити приблизний вигляд того, як може виглядати програма. Тут можна побачити структуру, яка спочатку отримує блоки програми, потім бере в них поля source і одні перетворює на текст, розділений на частини, а в інших отримує назви слайдів, щоб потім завантажити їх з сервера. Внизу можна побачити вікно, задача якого відображати те, що відбувається на головному екрані. Для того, щоб надіслати сигнал, який буде виводити на екран матеріал, потрібно двічі клікнути по блоку

пісні чи слайду і тоді воно відправиться на вивід до десктопного застосунку. Також, можна згорнути відформатований текст чи слайди до короткої лінії, яка буде відображати лише головну інформацію.

Якщо казати про структуру цього проєкту, то тут система не надто велика. Натомість, її структура будується наступним чином:

/src

/java

/models

Slide.java - модель слайду тексту чи зображення

SlideImage.java - модель слайду з зображенням

MainActivity.java - головний файл, з якого починається запуск

ServerWebSocket.java - клас, що відповідає за роботу з сервером

SlidesAdapter.java - клас, що відповідає за адаптацію слайдів

/res

/drawable

slide_item_bg.xml

text_slide_item_bg.xml

/layout

activity_main.xml - дизайн головної сторінки

item_slide.xml - дизайн елементу слайду

item_text_slide.xml - дизайн текстового слайду

AndroidManifest.xml

Відповідно, така невелика структура проєкту може забезпечити базові вимоги, представлені в задачах проєкту.

Побудова проєкту

Для програмування проєкту використовувалося середовище Android Studio, перевагою якого є можливість запускати програму як на попередньо встановленому симуляторі, так і на підключеному смартфоні.

Для того, щоб додати симулятор потрібно виконати наступні кроки:

Відкрити Tools → Device Manager

натиснути на + → Create Virtual Device

Вибрати пристрій, який буде симулюватися

Вибрати розмір оперативної пам’яті, кількість ядер та вбудованої пам’яті;

Для того, щоб запустити симулятор потрібно перейти до Running Device на панелі справа та вибрати раніше згенерований симулятор, після чого він запуститься. Після цього програму буде можливо запустити в режимі або Debug або Release.

ВИСНОВКИ ДО РОЗДІЛУ 2

В ході опрацювання другого розділу здійснено комплексне проєктування архітектури системи та розроблено всі чотири компоненти розподіленої системи дистанційного керування виводом мультимедійного контенту на екран.

Розроблено архітектуру, що складається з чотирьох взаємопов'язаних компонентів: серверного модуля, веб-інтерфейсу, десктопної програми та Android-застосунку. Спроєктовано структуру реляційної бази даних з п'ятьма основними таблицями та визначено формат JSON-файлів для зберігання блоків програми заходу. Створено UML-діаграми варіантів використання та схеми взаємодії компонентів.

Здійснено обґрунтований вибір технологічного стеку: Angular для веб-інтерфейсу, Qt C++ для десктопної програми, Java для Android-застосунку. Визначено протоколи взаємодії: WebSocket для синхронного керування в режимі реального часу, REST API для передачі структурованих даних, HTTP з MJPEG-кодуванням для трансляції відеопотоку екрану. Обрано PostgreSQL для серверного зберігання даних та SQLite3 для локального кешування.

Розроблено детальні інструкції з розгортання кожного компонента системи, включаючи списки необхідних бібліотек, команди встановлення залежностей та процедури компіляції. Створено CMakeLists.txt файли для автоматизації збірки C++-компонентів. Описано процес крос-компіляції для Raspberry Pi з використанням sysroot та ARM-компілятора.

Результати другого розділу демонструють успішну реалізацію всіх чотирьох компонентів системи з дотриманням встановлених технічних вимог щодо енергоефективності, продуктивності та зручності використання. Створена архітектура забезпечує гнучкість, масштабованість та можливість подальшого розвитку функціоналу системи.

РОЗДІЛ 3
ВИПРОБУВАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ

В цьому розділі головною задачею є випробування розробленої системи. Випробування розробленої системи потрібне для того, щоб можна було зрозуміти наступне:

1. підтвердити відповідність створеної системи тому, що було поставлено в рамках вимог;
2. виконати заміри ефективності програми та визначити, чи дійсно вона щось покращує;
3. оцінити рівень якості та надійності;
4. знайти помилки, які можуть виникнути в ході експлуатації програми, в тому числі, якщо застосунок буде використовуватись не за цільовим призначенням;

Перш ніж перейти до безпосереднього випробування системи, спочатку необхідно описати показники за якими буде порівнюватись нова система з попередньою.

3.1 Показники оцінювання ефективності

Якщо згадати головні проблеми альтернативних систем, то можна виділити наступні пункти, які є проблемними та оцінка того, що можна вважати покращенням:

1. формування програми заходу на робочому місці оператором екрану: як вже згадувалося раніше, для того, щоб сформувати програму, оператор мав прийти на робоче місце, відкрити канал, через який йому відправляють матеріали та додати їх до програми. Це вимагає певного часу, особливо коли замовники можуть не знати, в якому форматі краще відправляти матеріали. Задачею ж проєкту було те, щоб замовники самостійно додавали матеріал до

програми, що значно економило б час у оператора. Відповідно, першим показником ефективності є скорочення часу, який витрачає оператор на підготовку;

2. постійна прив’язка оператора екрану до робочого місця: це є ще однією проблемою, згаданою раніше. Відповідно, рішенням цієї проблеми є те, що оператор екрану повинен мати змогу керувати екраном з будь-якої точки, як мінімум, залу, в якому проводиться подія;

3. проблеми з перемиканням слайдів: суть цієї проблеми полягає в тому, що презентатори далеко не завжди можуть добивати сигнал до комп’ютера для того, щоб перемкнути слайди. Звичайно, спікер міг би перемикати слайди по wi-fi з телефона, але іноді це є проблемою, бо тоді за кафедрою потрібно тримати декілька пристроїв. Саме тому, рішенням цієї проблеми є те, що спікер зможе як бачити конспект своєї промови, так і перемикати потрібні йому слайди не просячи про це оператора екрану.

Після того, як були визначені показники оцінювання ефективності нової системи, тепер можна перейти безпосередньо до її оцінювання.

3.2 Оцінювання ефективності нової системи

Якщо звернутись до поставленого завдання, то першим показником, який потрібно перевірити, це на скільки менше часу оператор екрану використовує для розгортання системи виводу на екран та підготовки матеріалів до показу. Щоб оцінити цей пункт, потрібно оцінити скільки в середньому раніше оператор потребував часу, щоб все підготувати.

Підготовка матеріалу зазвичай ділилася на декілька етапів: підготовка пісень, підготовка слайдшоу, підготовка аудіо та відео. Відповідно, оцінимо скільки часу витрачається на підготовку одного елементу кожного типу медіа.

Таблиця 3.1

Розрахунок середньої витрати часу на підготування програми

Тип медіа	Тривалість підготовки одного елементу, хв	Середня кількість елементів на одну програму, од.	Тривалість підготовки загальної к-сті елементів в середньому, хв.
Пісня	3	4 - 5	12 - 15
Слайдшоу	3	1-2	3-6
Відео	2	1	2
Аудіо	2	1	2
Загалом:	10	7 - 9	19 - 25

Як можна побачити з таблиці вище, на підготовку однієї програми, оператор може витрачати до 25 хвилин повторюваної роботи. На противагу такій системі, створена в цій роботі система може взагалі не вимагати ніяких зусиль від оператора, щоб підготувати програму. Все основне навантаження розподіляється на тих, хто відповідає за різні блоки програми, а зазвичай це різні люди.

Наступним етапом випробувань є перевірка системи на те, чи можливо керувати виводом на екран віддалено за допомогою смартфона або планшета. Для того, щоб виконати цю задачу, було розроблено ще один застосунок під Android. Головною задачею цього застосунку є підключення до десктопного застосунку, отримання від нього програми заходу та відображення її у відповідному зручному вигляді, так, щоб можна було по маленьким блокам надсилати контент на вивід на екран, що зображено на рис. 3.1 Також значним

чином був доповнений новим програмним кодом десктопний застосунок. В ньому з’явилась можливість підключатися до нього за допомогою WebSocket, надсилати до нього команди, які він здатний розпізнавати та робити відповідні дії.

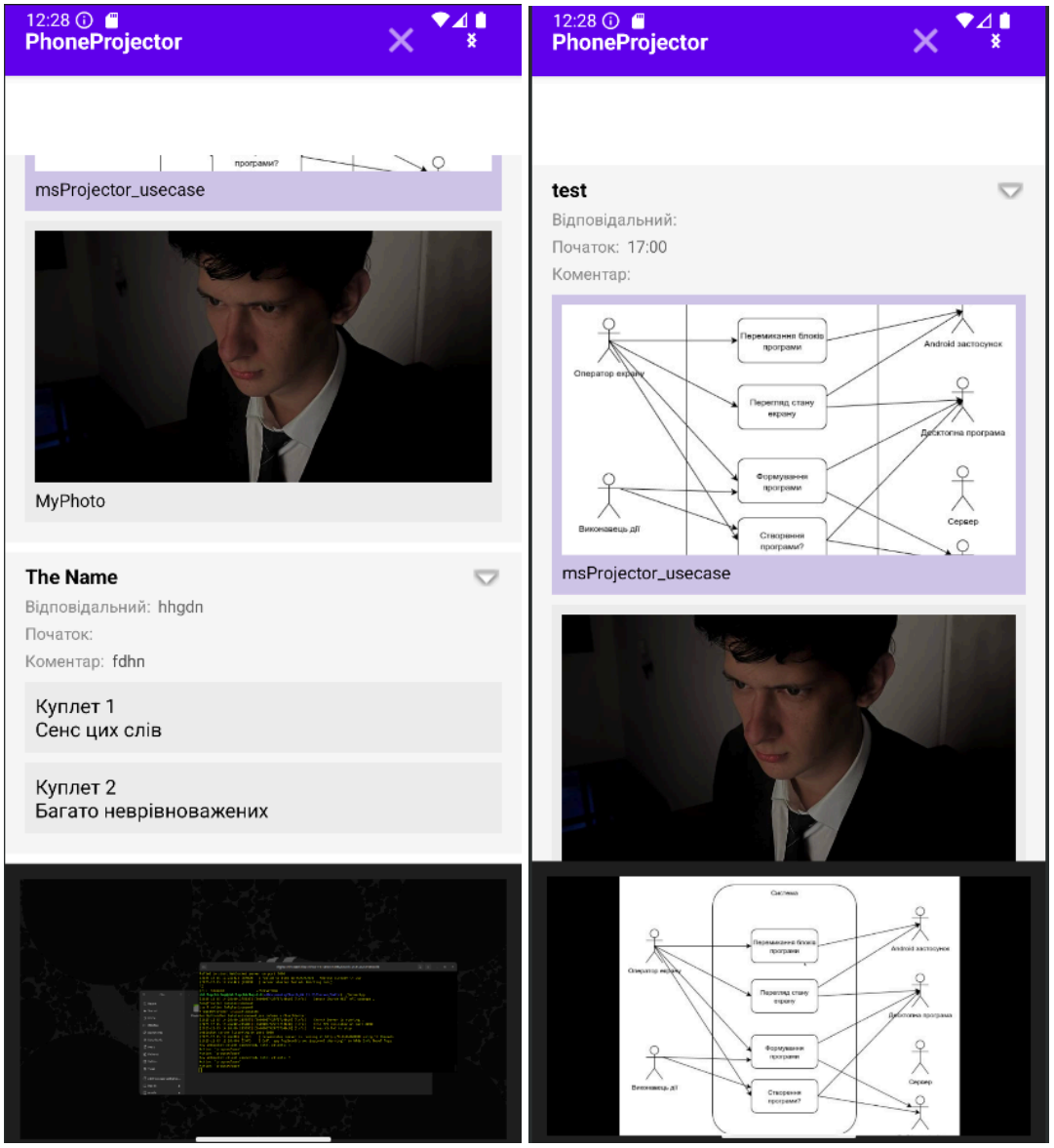

Рис. 3.1 – інтерфейс мобільного застосунку

На рисунку 3.1 зображено програму заходу, яка містить в собі слайди з їх назвою, та пісню, розділену на блоки для виведення. В нижній панелі програми відображається стан головного екрану в живому часі.

Останнім етапом випробувань системи є перевірка її на те, на скільки вона є зручною та інтуїтивно зрозумілою у порівнянні з її аналогами.

3.3 Експериментальне тестування

Для оцінки продуктивності розробленої системи розроблено програму, яка буде проводити навантажувальне тестування WebSocket–сервера, розгорнутого на одноплатному комп'ютері Raspberry Pi 3B. Тестування здійснювалося шляхом створення множинних одночасних з'єднань з сервером та вимірювання часу відповіді на керуючі команди. Параметри тестування включали від 1 до 10 одночасних підключень з інтенсивністю відправки команд від 100 до 1000 команд/сек відповідно. В таблицях 3.2 та 3.3 продемонстровано узагальнені результати тестування.

Таблиця 3.2

Показники продуктивності WebSocket-сервера

К-сть підключень	Відправлено команд	Отримано відповідей	команд / сек	середня затримка (мс)	успішність (%)
1	95	95	58.87	5.41	100
2	190	190	117.07	-5.19*	100
5	475	475	293.72	5.63	100
10	950	950	583.45	11.72	100

Примітка: Від’ємне значення пов’язане з особливостями синхронізації часу між клієнтом та сервером і не вказують на проблеми продуктивності.

В таблиці 3.3 можна побачити показники мінімальної та максимальної затримки відповіді зі сторони сервера під час надсилання на нього команд.

Таблиця 3.3

Розподіл затримок відповідей

Кількість підключень	Мін. затримка (мс)	Макс. затримка (мс)	Стандартне відхилення (мс)	Діапазон (мс)
1	-10.13	70.21	11.28	80.34
2	-10.34	2.45	1.67	12.79
5	-10.51	20.48	3.18	30.99
10	-10.80	71.90	12.01	82.70

Результат дослідження пропускної здатності серверної частини свідчить про лінійне зростання пропускної здатності сервера при збільшенні кількості одночасних підключень. Так, наприклад при одному підключенні система обробляє 58.87 команд на секунду, а при десяти підключеннях показник зростає до 583.45 команд на секунду, що свідчить про ефективне використання ресурсів.

Середня затримка відповіді залишається на прийнятному для людського ока рівні при будь-якому сценарії навантаження:

1. При 1 підключенні: 5.41 мс;
2. При 5 підключеннях: 5.63 мс;
3. При 10 підключеннях: 11.72 мс.

Збільшення середньої затримки при зростанні кількості підключень є очікуваним і пояснюється конкуренцією за процесорні ресурси одноплатного комп’ютера.

Згідно з характеристиками Raspberry Pi 3B, максимальна рекомендована потужність становить 5.1V/2.5A (12.75W), такий рівень споживання однозначно є найнижчим, адже інші системи для зберігання інформації локально використовують комп’ютери, які в середньому споживають від 150 до 300V, та

лептопи, які в середньому споживають від 30 до 90V . Відповідно, використання одноплатного комп’ютера є енергоефективним рішенням у порівнянні з іншими системами.

Варто відзначити показники ефективності десктопного застосунку. В таблиці 3.4 відображено показники споживання при використанні застосунку в звичному режимі.

Таблиця 3.4

Використання апаратних ресурсів десктопною програмою

Система	Назва компоненту	CPU, %	ОЗУ
Моя програма	Головне вікно	~ 0.7	180 – 230 Мб
	Транслятор екрану	~ 2.7	8 – 10 Мб
ProPresenter	Головне вікно	~ 20	700 – 800 Мб
FreeShow	Головне вікно	1	120 – 140 Мб

В таблиці вище можна побачити надзвичайно ефективну з точки зору апаратних ресурсів програму FreeShow, яка демонструє собою низьке споживання оперативної пам’яті при тому, що вона здатна демонструвати різні типи медіа матеріалу, включаючи відео потік. Це стало можливим за рахунок використання легкого фреймворку Svelte [21] та скриптової мови програмування JavaScript. Зважаючи на легкість роботи цих фреймворків з пам'яттю та оптимізованістю під різні системи, одним з напрямків розвитку проєкту може бути використання цих технологій для створення десктопного додатку.

ProPresenter є професійним комерційним рішенням, над розробкою якого працює команда фахівців. Але сама програма використовує дещо більше

апаратних ресурсів для виконання таких самих завдань як і FreeShow, тому її використання не є доцільним.

Система ж дистанційного керування виводом на екран має на меті дати можливість користувачу керувати виводом віддалено, з цієї точки зору система є достатньо ефективною, але її недоліком на даному етапі є досить обмежений функціонал та спектр типів медіа, з якими він може працювати. Саме тому можна звернутись до позитивного досвіду FreeShow та використовувати Svelte та JavaScript для розробки десктопної програми, функціонал якої буде розширено за рахунок синхронізації даних та можливості керувати презентацією віддалено.

3.4 Оцінка достовірності одержаних результатів

В ході розробки системи, яка б забезпечувала вирішення поставлених задач можна прослідкувати які саме задачі успішно виконані, а які вимагають додаткових досліджень.

З впевненістю можна сказати, що була вирішена задача, суть якої полягає в тому, щоб люди могли формувати програму дистанційно через веб–інтерфейс. Для повноцінного ж завершення цієї частини роботи необхідно десь запустити цей сайт та сервер до нього, щоб вони були в доступі віддалено, а не лише в одній локальній мережі. Для цього потрібно розгорнути сервер, на якому міг би крутитись сайт, придбати доменне ім’я для сайту та прив’язати його до нього. Так само і з сервером: йому потрібно або надати білу публічну ір адресу, через яку сайт зможе звернутися до нього, або також прив’язати доменне ім’я.

Якщо казати про перемикання слайдів дистанційно, то тут дійсно можна виконувати надсилання сигналів на комп’ютер, який вже буде виконувати відповідні дії. Однак, є проблема, яка полягає в тому, що моя система є не достатньо функціональною з тієї точки зору, що вона може виводити лише текст

та зображення, а для конкурентоспроможності їй бажано було б мати можливість виводити ще хоча б відео на екран та програвати аудіо.

Останнім завданням, яке частково входить в попередній пункт, полягає в тому, щоб спікер міг перемикати презентацію через телефон або планшет. Ця проблема частково вирішена за рахунок того, що йому можна просто передати відповідний пристрій, на якому він зможе виконувати відповідні дії.

3.5 Проблеми, що виникли в ході розробки проєкту

В ході розробки проєкту виникло чимало проблем з реалізацією, які варто згадати.

Першою проблемою є використання Qt C++ для написання десктопної програми. Сама мова C++ є досить не простою мовою, використовуючи яку в ході розробки важливо звертати увагу на виділення пам’яті під дані та їх коректне видалення. Також, програмі на C++ досить важко обробляти великий список слайдшоу, коли в кожному може бути до декількох десятків слайдів. Відповідно, проблеми можуть виникати також і у роботі з великим масивом нелегких відео. Одним із теоретичних рішень такої проблеми є, наприклад, використання іншої, більш сучасної та динамічної мови програмування, наприклад Java або C#, оскільки там автоматично використовуються механізми оптимізації та роботи з пам’яті, чого не скажеш про C++, в якому про це потрібно піклуватися самостійно. Або ж, можна запропонувати ще одну концепцію, в якій практично не використовується десктопна програма для виведення на другий екран. Така програма могла б бути практично консольною, а редагування програми та виведення того чи іншого контенту лягало б на веб–інтерфейс та / або на планшетну програму, яка б просто підключалась до цієї консолі і відправляла туди відповідні команди на виведення тієї чи іншої інформації. Це позбавило б необхідності думати про те, як написати такий оптимізований код для виведення великого списку зображень чи відео і

дозволило б програмі читати і виводити, наприклад, відео безпосередньо з директорії.

В ході розробки також була виявлена ще одна проблема, це використання бази даних SQLite 3. Проблема в тому, що в нашій концепції використовується сервер, який через RESTful API отримує певну інформацію та зберігає її у файловій базі даних. І також в нас є десктопна програма, яке робить те ж саме локально. І проблема тут може виникнути в ході синхронізації, тому що читання та запис одночасно у файл може призвести до збою та втрати даних. Відповідно, до цього, було вирішено використовувати базу даних на основі postgresql. Її основною перевагою є те, що вона вже являє собою такий собі сервер, до якого можна підключитись через адресу та порт і напряму витягнути з неї потрібну інформацію, як от слайди, тексти, відео тощо, економлячи час на розробку додаткового RESTful API сервера та уникаючи ризиків, пов’язаних з синхронізацією даних. Тим не менш, щоб мати доступ до бібліотеки матеріалів, RESTful API сервер все рівно потрібно буде розробити, оскільки фреймворк angular не містить в собі бібліотек, які могли б напряму звертатися до postgresql бази даних, для цього йому потрібен проміжний етап. Також, якщо буде використовуватись база даних, доступ до якої є через мережу, тоді потрібно забезпечити стабільним зв’язком, бо якщо він зникне, або виникнуть певні проблеми, програма просто не зможе нічого виводити на екран, бо в неї не буде ані бази пісень, ані слайдів. Частково, є ще одні рішення цієї проблеми, це спробувати синхронізувати локальну базу даних з серверною. І в випадку якщо зникне зв’язок, десктопна програма зможе працювати автономно не залежачи в цьому випадку від сервера.

Останньою основною проблемою є обмеженість ресурсів. В ході використання міні комп’ютера raspberry pi 3 як сервера виникла проблема з нестачею ресурсів. Оскільки цей міні комп’ютер побудований на базі процесора arm, зкомпілювати виконуваний файл сервера під нього можна лише відповідно на такому ж самому процесорі. Але поточний міні комп'ютер не містить

відповідних достатніх потужностей, через що він не може скомпілювати проєкт, який містить і реалізацію Websocket сервера, і RESTful API сервера. Відповідним ймовірним рішенням цієї проблеми є використання новішої версії цієї лінійки, яка могла б містити хоча б 2-4 Гб оперативної пам’яті, замість 1 Гб в цій версії, та від 1.5 до 2.4 ГГц у процесорі замість 1.2 ГГц в поточного.

3.4 Перспективи розвитку проєкту

Якщо поглянути загалом на всю систему, то в ній можна помітити такі недоліки, які можна покращити. З урахуванням нестачі часу, можливості реалізувати повний функціонал немає. Спланованої подальші кроки можуть збільшити зручність системи та зекономити інтелектуальні ресурси людини.

Першим покращенням, є зменшення кількості компонентів, які задіяні в системі. Ця система містить два схожі компоненти - сервер та десктопна програма. Є складність того, що між ними потрібно здійснювати синхронізацію бази даних, що не є рекомендованим, оскільки використовується файлова база даних. Рішення полягає в тому, щоб можна було знайти спосіб, завдяки якому сервер сам міг би отримувати команди, знаходити другий екран та виводити на нього те, що йому передали. В такому випадку, програму можна було б створювати та редагувати у веб–інтерфейсі. Відповідно, можна створити застосунок під планшет з ширшим функціоналом, що вичерпало б необхідність десктопної програми. Вся робота може здійснюватись через планшет, який відкриває програму, вивантажуючи її з міні комп’ютера raspberry pi.. Але для реалізації цього рішення необхідно знайти технології, які можуть поєднувати в собі сервер, який працює завжди, і графічну частину, яка бере дані з серверної бази та працює з ними.

Як можна було побачити, для створення програми заходу дистанційно використовується окремий веб–інтерфейс. Його проблема може бути в тому, що до цього ресурсу можуть отримати доступ не валідовані люди. Щоб цьому

запобігти, потрібно реалізувати систему аутентифікації та авторизації, що вимагає додаткової розробки і системи контролю доступу до тих чи інших даних. Можливим рішенням цієї системи може бути створення міні застосунку, яке вбудоване в соцмережі телеграм. В такому випадку, щоб не створювати свою власну систему авторизації та перевірки людей, можна надати дозвіл певним користувачам, які зможуть бачити та створювати програми всередині застосунку.. В такому міні додатку можна буде реалізувати створення нових програм, редагування вибраної, додавання туди медіа матеріалів, що відмінить необхідність реалізації системи аутентифікації.

Останнім з напрямів подальшого розвитку системи є реалізація функціональності інтеграції веб-ресурсів безпосередньо в структуру програми заходу. У поточних програмних рішеннях відображення онлайн-контенту здійснюється шляхом деактивації режиму дублювання екранів та ручного перенесення вікна браузера на проєкційний дисплей, що порушує цілісність керування презентацією. Пропонується впровадження модуля рендерингу веб-сторінок, який забезпечить вбудоване відображення інтернет-ресурсів.

ВИСНОВКИ ДО РОЗДІЛУ 3

В ході третього розділу проведено комплексне випробування розробленої системи дистанційного керування виводом на екран, що підтвердило досягнення поставлених цілей та виявило напрямки подальшого вдосконалення.

Встановлено три ключові показники для оцінювання ефективності системи: скорочення часу підготовки медіаматеріалів оператором, забезпечення можливості дистанційного керування з будь-якої точки залу, вирішення проблеми перемикання слайдів спікерами без залучення оператора.

Проведений аналіз показав значну економію часу на підготовку програми заходу. У традиційному підході оператор витрачає від 19 до 25 хвилин на підготовку однієї програми. Розроблена система повністю усуває цю повторювану роботу, перенісши її на організаторів через веб-інтерфейс, що дозволяє оператору зосередитися на технічних аспектах проведення заходу.

Проведено серію експериментальних тестів продуктивності серверної частини на Raspberry Pi 3B при різних рівнях навантаження. Результати підтвердили лінійне зростання пропускної здатності: від 58.87 команд/сек при одному підключенні до 583.45 команд/сек при десяти підключеннях. Середня затримка відповіді залишається на прийнятному рівні: 5.41 мс, 5.63 мс, 11.72 мс. Максимальна зафіксована затримка становить 71.90 мс при 10 підключеннях, що значно нижче порогу помітності для людського ока. Відсоток успішних відповідей у всіх сценаріях становить 100%, що свідчить про високу надійність системи.

Підтверджено енергоефективність рішення: споживання Raspberry Pi 3B не перевищує 12.75 Вт (5.1V/2.5A), що є в 12-24 рази ефективніше порівняно з традиційними серверними рішеннями (150-300 Вт) та в 2-7 разів ефективніше ноутбуків (30-90 Вт). Десктопна програма споживає 180-230 МБ ОЗП та ~0.7% CPU, що є конкурентоспроможним показником.

Підтверджено успішну реалізацію основних функціональних вимог: організатори можуть формувати програму заходу дистанційно через веб-інтерфейс без участі оператора; оператор може керувати виводом контенту з будь-якої точки конференц-залу через Android-застосунок; спікери можуть перемикати слайди через мобільний пристрій без звернення до оператора; система забезпечує візуальний контроль відображуваного контенту через трансляцію екрану.

Результати третього розділу підтверджують технічну життєздатність розробленої системи та її відповідність встановленим вимогам щодо продуктивності, надійності та енергоефективності. Виявлені обмеження є типовими для MVP-версії продукту та мають чіткі шляхи вирішення в наступних ітераціях розробки. Система демонструє конкурентоспроможні показники порівняно з існуючими рішеннями та має потенціал для подальшого вдосконалення.

РОЗДІЛ 4
РОЗРОБКА СТАРТАП-ПРОЄКТУ

4.1 Опис ідеї

Проєкт системи для дистанційного керування виводом на екран покликаний дати можливість виводити певний медіа контент на екран не знаходячись на певному конкретному місці використовуючи комп’ютер. В певній мірі, однією з задач системи є можливість розподілити формування програми між усіма зацікавленими учасниками того чи іншого заходу, тим самим позбавити оператора екрану зайвої, не потрібної, повторюваної роботи. Відмінними характеристиками проєкту є:

1. формування медіа матеріалів для певного заходу через веб–інтерфейс, відповідно в будь-якому місці, не встановлюючи при цьому ніякого додаткового програмного забезпечення;
2. керування виводом на екран віддалено через смартфон або планшет.

Унікальність розробки полягає в широкому спектрі використовуваних технологій для забезпечення зручності для кінцевих користувачів. Вся ця система пов’язана між собою за допомогою різноманітних мережевих протоколів, кожен з яких бере на себе різну задачу. Загальний опис ідеї стартап-проєкту можна знайти в таблиці 4.1

Таблиця 4.1

Опис ідеї стартап-проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Веб–інтерфейс для формування програми заходу	Організації, де є програма заходу супроводжувана медіаконтентом	Організатори формують через веб–інтерфейс.. Оператор екрану не робить рутини
Застосунок керування віддаленим виведенням	оператори керування виводом медіа контенту на екран	Оператор керує з будь-якого місця в залі. Спікер / ведучий перемикає слайди не звертаючись до оператора

Основними аналогами проєкту є інше програмне забезпечення, задача якого є виведення матеріалу на екран. Порівняльний аналіз потенційних техніко-економічних переваг ідеї проєкту порівняно з конкурентами наведено у таблиці 4.2

Таблиця 4.2

Визначення сильних, слабких та нейтральних характеристик ідеї проєкту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари / концепції конкурентів			
		Мій проєкт	ProPresenter	Free Show	Open LP
1.	Наявність керування через смартфон	+	+	+	-
2.	Можливість формування програми віддалено	+	-	-	-

Продовження таблиці 4.2

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари / концепції конкурентів			
		Мій проєкт	ProPresenter	Free Show	OpenLP
3.	Легкість в розгортанні системи	-	+	+	-

4.2 Технологічний аудит проєкту

Розробка системи для дистанційного керування виводом на екран потребує аудиту технологій, за допомогою яких можна реалізувати такий проєкт. Це необхідно для того, щоб виявити можливі технічні проблеми, з якими можна зіткнутися в процесі впровадження. Важливо сконцентрувати увагу на стабільності мережевих зв’язків між компонентами, а також навантаженості системи, яка буде відправляти різні медіа матеріали на екран. В таблиці 4.3 можна побачити аналіз того, на скільки можливим є створення задуманої системи.

Таблиця 4.3

Технологічна здійсненність ідеї проєкту

№ п/п	Ідея проєкту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Веб–інтерфейс для взаємодії з сервером	Фреймворк для створення веб–інтерфейсів Angular	Наявний	Загальний безкоштовний доступ

Продовження таблиці 4.3

№ п/п	Ідея проєкту	Технології її реалізації	Наявність технологій	Доступність технологій
2	Зв’язок сайту з сервером для доступу до бібліотеки матеріалів	RESTful API та відповідні бібліотеки у фреймворці Angular	Наявний протокол та бібліотеки	Загальнодоступна технологія
3	Зв’язок сайту з сервером для живого формування програми заходу	Web Socket протокол та відповідні бібліотеки у Angular	Наявний протокол та бібліотеки	Загальнодоступний протокол та бібліотеки
4	Сервер для зберігання матеріалів та синхронізації програми заходу	C++, Web Socket Server, RESTful API обробник, SQLite 3 / Postgresql	Наявна мова програмування та протоколи	Загальнодоступні технології
5	Безпосередній вивід медіа матеріалів на екран	Бібліотека Qt та можливість відслідковування додаткових екранів	Наявна	Безкоштовно доступна для ОС Linux, для Windows - за ліцензією
6	Відправлення команд на вивід матеріалів з телефону	Web Socket протокол для зв’язку з застосунком	Наявний протокол	Загальнодоступна технологія

Обрані технології для реалізації проєкту включають такі фреймворки / мови програмування як C++, Java та Angular, а також технології / протоколи як от Web Socket, RESTful API, SQLite3 / Postgresql. Всі відповідні програми, написані на різних мовах програмування будуть поєднуватись один між одним за допомогою різноманітних мережевих технологій, а дані всі дані будуть зберігатись на сервері у базі даних SQLite 3 / Postgresql.

Загалом, можна сказати, що реалізація задуму є можливим, оскільки існують технології, які можуть забезпечити виконання вказаних вимог. Необхідні для цього технології існують, є загальнодоступними що дозволяє реалізувати всі аспекти цього проєкту.

4.3 Аналіз ринкових можливостей запуску

Наступним етапом у реалізації стартап-проєкту є визначення сприятливих обставин для запуску стартап-проєкту системи дистанційного керування виводом на екран. Це важливо для розуміння того, як та з чим потрібно виходити на ринок та отримання конкурентних переваг. Розуміння ринку дозволяє ефективно планувати розвиток проєкту, виявляти потреби цільової аудиторії та оцінювати ризики. В таблиці 4.4 можна побачити характеристику потенційного ринку для цього стартап-проєкту.

Таблиця 4.4

Попередня характеристика потенційного ринку стартап-проєкту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	5

Продовження таблиці 4.4

№ п/п	Показники стану ринку (найменування)	Характеристика
2	Динаміка ринку (якісна оцінка)	Помірна
3	Наявність обмежень для входу (вказати характер обмежень)	Відсутні
4	Специфічні вимоги до стандартизації та сертифікації	Відсутні
5	Середня норма рентабельності в галузі (або по ринку), %	20% (експертна оцінка)

Початковий аналіз показує, що ринок є відносно сталим для впровадження продукту. Відсутність специфічних обмежень та досить не широка конкуренція сприяє позитивному середовищу для запуску проєкту.

В таблиці 4.5 наводиться характеристика потенційних клієнтів, включаючи їхні основні потреби та вимоги, що формує ринковий попит на програмний продукт.

Таблиця 4.5

Характеристика потенційних клієнтів стартап-проєкту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Вимоги споживачів до товару
1	Економія людського ресурсу на виконанні повторюваної роботи	Оператори екрану, організатори заходу	Стабільність системи

Продовження таблиці 4.5

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Вимоги споживачів до товару
2	Можливість працювати не прив’язуючись до конкретного робочого місця	Оператори екрану	Надійність мережі для забезпечення стабільної передачі команд
3	Економія комп’ютерних потужностей	Організація	Економічність ПЗ у використанні електромережі

Цільовою аудиторією проєкту є оператори екрану, задача яких є виведення тих чи інших медіа матеріалів під час заходу на екран. Серед їхніх основних вимог є стабільність системи, надійність мережевого з’єднання між різними компонентами, відмовостійкість при виникненні критичних ситуацій.

Втім, для успішної реалізації проєкту потрібно врахувати можливі загрози, які можуть виникнути в ході енергетичної ситуації, мережевої, або ж в ході появи нових конкурентів на ринку. Це дозволить ефективно оцінювати зміни на ринку та забезпечувати стійкість проєкту в лиці потенційних загроз. Відповідно, у таблиці 4.6 можна знайти фактори загроз, що можуть перешкоджати ринковому впровадженню.

Таблиця 4.6

Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Поява нових конкурентів	Конкуренти, що запропонували більш ефективне рішення	Продумана візія компанії, уявлення про те, який стан системи є ідеальним та поступовий шлях до цього ідеалу.
2	Енергетична нестабільність	Часте вимикання світла, часом непередбачуване	Розробка рішень альтернативних джерел енергії
3	Мережева нестабільність	Проблеми з інтернетом через відсутність електрики у в районі	Розробка схем розбудови локальної мережі, яка працювала б в умовах конкретного приміщення

У таблиці 4.7 можна побачити фактори можливостей, що сприяють ринковому впровадженню.

Таблиця 4.7

Фактор можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Вихід системи на міжнародний рівень	Систему можна розширити на весь світ	Надання можливості в системі реєструвати

Продовження таблиці 4.7

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
		для того, щоб в ній могли реєструватись різні організації та формувати там свої програми заходів	організації, де буде можливість формувати програму та керувати нею
2	Орієнтованість системи на заміну людини	Покращення системи до рівня використання ШІ, який міг би орієнтуватися в тому, що відбувається на заході і відповідним чином відтворювати той чи інший контент	Розробка та залучення різних моделей ШІ в програмному забезпеченні, яке може орієнтуватись на основі відео з камер та звуці з мікрофонів, для того щоб приймати рішення

У таблиці 4.8 наведено загальні риси конкуренції на ринку.

Таблиця 4.8

Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Тип конкуренції: чиста	конкуренція між розробниками систем для керування екраном	Поширення системи на інші країни світу, запровадження нових інноваційних рішень в цій системі

Продовження таблиці 4.8

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
За рівнем конкурентної боротьби: глобальний	Якість системи не залежить від країни чи локалізації	Впровадження різних бібліотек для взаємодії з різними форматами файлів та даних.
За галузевою ознакою: внутрішньогалузева	Усі продукти є програмними	Додавання функцій, які б дозволяли редагувати матеріал, який додається до заходу
Конкуренція за видами товарів: між бажаннями	Полягає у задоволенні потреб операторів екрану та учасників організацій ефективніше за конкурентів	Проведення зворотного зв’язку з користувачами та внесення поліпшень в систему
За характером конкурентних переваг: нецінова	Задоволення потреб користувача через розширення функцій	Додавання нових функцій, мов, каналів підтримки

У таблиці 4.9 наведено більш детальні риси конкуренції на ринку за М. Портером.

Таблиця 4.9

Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Інші застосунки для виведення матеріалу на екран та керування ним дистанційно	Інші стартапи, які можуть пропонувати подібні рішення з простішою взаємодією	Постачальники електроенергії та інтернету	Оператори екранів, організатори заходів	Інші продукти, які дозволяють виводити текст / зображення на другий екран
Висновки:	На ринку є інше ПЗ для виведення на другий екран, включаючи і можливість дистанційного керування	Потенційні конкуренти мають рішення для виведення медіа на другий екран та керування ним з телефону	Не диктують умови на ринку, якщо вчасно оплачувати їхні послуги	Клієнти диктують вимоги щодо зручності системи та її функціональності	Конкуренти мають значні переваги в

На основі аналізу конкуренції, а також із урахуванням характеристик ідеї проєкту, вимог споживачів та факторів маркетингового середовища, в таблиці 4.10 визначено та обґрунтовано перелік факторів конкурентоспроможності.

Таблиця 4.10

Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проєктів значущим)
1	Енергоефективність	Дозволяє виконувати всі основні запити не використовуючи для цього значних обчислювальних ресурсів, та, відповідно, енергетичних
2	Цінова політика	Вартість розгортання системи робить його доступним для невеликих та середніх організацій
3	Розповсюдження системи на різні країни	Система передбачає можливість застосування в різних країнах, організаціях, забезпечуючи потреби користувачів по всьому світу

Наступним етапом ринкового аналізу можливостей є складання SWOT-аналізу на основі сильних та слабких сторін. Відповідний аналіз сформовано в таблиці 4.11.

Таблиця 4.11

SWOT-аналіз стартап-проєкту

Сильні сторони	Слабкі сторони
Споживання сервера не більше 5 Вт, десктопної програми до 230 Мб, формування програми заходу через веб–інтерфейс	Відсутність підтримки широкого кола медіа форматів, складність розробки десктопної програми

Продовження таблиці 4.11

Можливості	Загрози
Зростаючий попит на економію інтелектуального ресурсу шляхом скорочення повторюваної роботи та запит на свободу руху	Конкурентоспроможні компанії, які зможуть придумати більш інноваційні рішення, або запропонувати більш гнучку систему. Відсутність інтернету позбавить систему будь-якої унікальності

На основі SWOT-аналізу, в таблиці 4.12 відображено альтернативи ринкового впровадження стартап-проєкту.

Таблиця 4.12

Альтернативи ринкового впровадження стартап-проєкту

Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
Розробка функціоналу для керування звуком, світлом та іншими програмними ресурсами	Висока	9 місяців
Вихід на міжорганізаційний рівень	Висока	8 місяців
Перехід на рівень керування виводом використовуючи AI	Середня	2 роки

Для початкового запуску було обрано альтернативу з виходом на міжорганізаційний рівень.

4.4 Розроблення ринкової стратегії проєкту

Для визначення ринкової стратегії спочатку потрібно визначити стратегії охоплення ринку. В таблиці 4.13 можна побачити відповідний вибір цільових груп потенційних споживачів.

Таблиця 4.13

Вибір цільових груп потенційних споживачів

Опис профілю цільової групи потенційних клієнтів	Готовність споживачів прийняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
Організації, де відбуваються заходи	середня	середній	середня	середня
Навчальні заклади	низька	низький	не висока	складна
Концертні приміщення	висока	високий	висока	складна
Які цільові групи обрано: організації де регулярно відбуваються заходи, в яких регулярно залучається медіа матеріал та приміщення, в яких можуть проходити концерти, семінари та інші заходи.				

Наступним кроком необхідно сформувати базову стратегію розвитку. Відповідне визначення описано в таблиці 4.14.

Таблиця 4.14

Визначення базової стратегії розвитку

Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Розробка системи з базовим функціоналом	Диференцій ований маркетинг	Легкість в керуванні, інтуїтивно зрозумілий інтерфейс, можливість працювати віддалено	Диференціація

Після визначення стратегії потрібно вибрати стратегію конкурентної поведінки, яка описана в таблиці 4.15.

Таблиця 4.15

Визначення базової стратегії конкурентної поведінки

Чи є проєкт “першопро хідцем” на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
Ні	Компанія буде забирати існуючих у конкурентів	Компанія буде впроваджувати нові характеристики для взаємодії з іншими медіа сервісами	Стратегія заняття конкурентної ніші

На основі вимог споживачів обраних сегментів розроблена стратегія позиціонування, яка описана в таблиці 4.16, суть якої полягає у формуванні ринкової позиції, за яким споживачі мають ідентифікувати торгівельну марку або проєкт.

Таблиця 4.16

Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проєкту	Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту (три ключових)
1	Відмовостійкість	Стратегія диференціації	Використання фреймворків, які забезпечують надійність системи та обробників помилок	Надійність, обробка помилок, резервування даних
2	Широкий спектр функцій	Стратегія спеціалізації	Використання різноманітних бібліотек для взаємодії з різними форматами медіа	Функціональність, підтримка медіа форматів, інтуїтивно зрозумілий інтерфейс
3	Стабільність роботи через мережу	Стратегія диференціації	Використання надійних протоколів для взаємодії компонентів один між одним	Гнучкість, незалежність, швидкість

Ця стратегія позиціонування забезпечить зв’язок між потребами замовників та ключовими перевагами системи для віддаленого керування виводом на екран, акцентуючи увагу на його надійності, функціональності та інтуїтивно зрозумілому інтерфейсі.

4.5 Розроблення маркетингової програми стартап–проєкту

Для формування маркетингової концепції продукту необхідно спершу провести детальний аналіз його конкурентоспроможності. У таблиці 4.17 відображено ключові переваги концепції продукту .

Таблиця 4.17

Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Формування програми віддалено	Веб–інтерфейс, на якому можна створити програму та редагувати її	Не потрібно завантажувати додаткове програмне забезпечення
2	Керування виводом з будь-якого місця	Система, в якій телефон підключається до комп’ютера та здійснює керування	Немає необхідності працювати з одного робочого місця

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення. Відповідний опис можна побачити в таблиці 4.18.

Таблиця 4.18

Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Глибина каналу збуту	Оптимальна система збуту
Оплата рекламної кампанії	Веб–інтерфейс, форуми, рекомендації між користувачами	Форум з інтеграцією рекомендацій користувачів

Однією з складових маркетингової програми є розробка концепції маркетингових комунікацій, що спирається на основу для позиціонування, обрану раніше. Відповідний опис можна побачити в таблиці 4.19.

Маркетингова програма спрямована на створення сильної брендової позиції системи для дистанційного керування виводом на екран як надійного, гнучкого, інтуїтивно зрозумілого інтерфейсу, а також інструменту, завдяки якому можна сформувати програму віддалено, додавши до неї потрібні медіа матеріали.

Таблиця 4.19

Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
Технічні форуми, GitHub	Соціальні мережі, технічні форуми	Технічна підтримка, актуальність	Показати переваги системи	Система дозволяє формувати програму заходу та керувати виводом віддалено

ВИСНОВКИ ДО РОЗДІЛУ 4

В ході опрацювання четвертого розділу проведено комплексний аналіз комерційного потенціалу стартап-проєкту системи дистанційного керування виводом на екран та розроблено стратегію виходу на ринок.

Сформульовано унікальну цінність продукту, що полягає у двох ключових можливостях: дистанційне формування програми заходу через веб-інтерфейс без встановлення додаткового ПЗ та керування виводом контенту через мобільні пристрої з будь-якої точки залу. Підтверджено технічну можливість реалізації всіх аспектів проєкту з використанням сучасного технологічного стеку. Виявлено 5 головних гравців, динаміка ринку оцінена як помірна, обмеження для входу відсутні, специфічні вимоги до сертифікації відсутні, середня норма рентабельності оцінена на рівні 20%. Визначено три цільові аудиторії з їхніми потребами: оператори екранів , організатори заходів, організації.

Визначено три альтернативи ринкового впровадження з різними строками реалізації: розробка розширеного функціоналу для керування звуком/світлом, вихід на міжорганізаційний рівень, впровадження AI-керування. Для початкового запуску обрано стратегію виходу на міжорганізаційний рівень.

Сформовано стратегію охоплення ринку з фокусом на три цільові сегменти: організації з регулярними заходами, навчальні заклади, концертні приміщення. Обрано базову стратегію диференціації з акцентом на легкість керування, інтуїтивний інтерфейс та можливість віддаленої роботи.

Визначено стратегію конкурентної поведінки як "заняття конкурентної ніші" шляхом залучення існуючих користувачів конкурентів через впровадження нових характеристик для інтеграції з медіасервісами.

Визначено дві ключові переваги концепції продукту: веб-інтерфейс для віддаленого формування програми без завантаження ПЗ та система

дистанційного керування через телефон без прив'язки до робочого місця. Сформовано систему збуту на базі веб-інтерфейсу, технічних форумів та рекомендацій користувачів, що забезпечує органічний ріст бази користувачів.

Розроблено концепцію маркетингових комунікацій з використанням технічних форумів, GitHub та соціальних мереж як основних каналів. Ключові позиції для позиціонування: технічна підтримка, актуальність рішення, економічна доступність. Завдання рекламного повідомлення - демонстрація переваг використання системи для різних цільових груп. Концепція звернення підкреслює можливість формування програми віддалено та дистанційного керування виводом.

Результати четвертого розділу демонструють комерційну життєздатність стартап-проєкту та підтверджують наявність ринкового попиту на розроблену систему. Розроблена стратегія виходу на ринок враховує специфіку цільових аудиторій, конкурентне середовище та фінансові можливості потенційних клієнтів. Визначені альтернативи розвитку та маркетингова програма створюють основу для успішної комерціалізації продукту.

ВИСНОВКИ

Завданням магістерської дисертації є розробка системи, яка позбавить оператора екрану виконувати повторюваної дії для підготовки медіа матеріалу для заходу та дозволить керувати виводом на екран з будь-якої точки конференц-залу використовуючи мобільний пристрій. В ході аналізу та порівняння існуючих рішень виявлено недоліки, серед яких є: незручний інтерфейс, оператору потрібно самостійно формувати програму з матеріалів, оператор прив’язаний до одного робочого місця, спікер використовує презентатор, сигнал якого може не досягати до робочої станції.

На противагу ж цим проблемам було запропоновано інноваційне рішення для економії інтелектуального ресурсу шляхом оптимізації виконуваної роботи за допомогою такої системи, яка б здійснювала реалізацію поставлених цілей та задач. Для цього здійснено детальне проєктування архітектури системи, сформовано типи даних, які використовуються для зберігання та відображення тих чи інших форматів даних. На основі описаних діаграм та схем розроблено систему, яка поєднує в собі 4 елементи - сервер, веб–інтерфейс, десктопна програма та android застосунок. Така система задовольняє виконання всіх задач, які були поставлені в першому розділі.

Проведено випробування ефективності створеної системи та перевірка досягнення поставлених цілей. Проведено функціональне тестування розробленої системи, в умовах, наближених до реальних сценаріїв використання. Експериментальна перевірка підтвердила працездатність функціональних модулів та досягнення поставлених цілей щодо автоматизації процесу підготовки програми, забезпечення дистанційного керування виведенням контенту, усунення залежності від традиційних презентаторів.

Наукова новизна роботи полягає у запропонованій архітектурі для дистанційного керування виводом візуальної інформації на екраном шляхом

використання протоколу WebSocket для передачі керуючих команд у режимі реального часу та протоколу HTTP з MJPEG-кодуванням для візуального контролю. Рішення забезпечує швидкість виконання команд на рівні з традиційним рішенням при використанні мережі. Для виконання роботи розроблено програмно-апаратний комплекс на базі одноплатного комп’ютера з оптимізованим серверним компонентом для організації програми заходу на основі JSON–файлів із централізованим зберіганням медіа матеріалів у базі даних PostgreSQL. Комплекс є енергоефективним і вимагає живлення на рівні 5 Вт при обслуговування чотирьох з’єднань, що означає, що з цією задачею справиться повербанк ємністю 10 000 mAh протягом приблизно чотирьох годин.

Недоліком розробленого рішення є нездатність працювати з таким широким колом медіа, як це вміють альтернативні дороговартісні програми. Згідно з цим напрямами подальшого розвитку проєкту визначено: розширення підтримуваних форматів медіаконтенту та оптимізація архітектури системи з метою зменшення кількості компонентів.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ProPresenter: The #1 Choice in Presentation Software [Електронний ресурс] / Renewed Vision. – Режим доступу: https://www.renewedvision.com/propresenter. – Дата звернення: 03.09.2025.
2. What Are the Minimum System Requirements for ProPresenter 7 [Електронний ресурс] / Scribd. – Режим доступу: https://www.scribd.com/document/822213858/What-are-the-minimum-system-requirements-for-ProPresenter-7-Renewed-Vision. – Дата звернення: 03.09.2025.
3. OpenLP [Електронний ресурс]. – Режим доступу: https://openlp.org/. – Дата звернення: 04.09.2025.
4. OpenLP – Minimum System Requirements [Електронний ресурс]. – Режим доступу: https://manual.openlp.org/system_requirements.html. – Дата звернення: 04.09.2025.
5. EasyWorship [Електронний ресурс]. – Режим доступу: https://www.easyworship.com/. – Дата звернення: 04.09.2025.
6. System Requirements for EasyWorship [Електронний ресурс]. – Режим доступу: https://support.easyworship.com/support/solutions/articles/6000047711-system-requirements-for-easyworship. – Дата звернення: 04.09.2025.
7. FreeShow [Електронний ресурс]. – Режим доступу: https://freeshow.app/. – Дата звернення: 05.09.2025.
8. FreeShow – GitHub issue #1024 [Електронний ресурс]. – Режим доступу: https://github.com/ChurchApps/FreeShow/issues/1024. – Дата звернення: 05.09.2025.
9. Overview – Angular [Електронний ресурс]. – Режим доступу: https://angular.dev/overview. – Дата звернення: 30.09.2025.

10. Qt Documentation [Електронний ресурс]. – Режим доступу: https://doc.qt.io/. – Дата звернення: 23.09.2025.
11. Developers – Android [Електронний ресурс]. – Режим доступу: https://developer.android.com/develop. – Дата звернення: 07.10.2025.
12. WebSockets API [Електронний ресурс] / Mozilla MDN. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API. – Дата звернення: 23.09.2025.
13. RESTful API: що це, як працює, переваги і приклади [Електронний ресурс] / GoIT Global. – Режим доступу: https://goit.global/ua/articles/rest-api-shcho-tse-iak-pratsiuie-perevahy-i-pryklady/. – Дата звернення: 23.09.2025.
14. HTTP MJPEG Video Streaming Implementation Guide [Електронний ресурс]. – Режим доступу: https://www.visioforge.com/help/docs/dotnet/general/network-streaming/http-mjpeg/ . – Дата звернення: 07.10.2025.
15. SQLite – About [Електронний ресурс]. – Режим доступу: https://sqlite.org/about.html. – Дата звернення: 23.09.2025.
16. PostgreSQL [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/PostgreSQL. – Дата звернення: 23.09.2025.
17. diagrams.net [Електронний ресурс]. – Режим доступу: https://app.diagrams.net/. – Дата звернення: 16.09.2025.
18. The Complete C++ Build System: CMake & Ninja (Part 1) [Електронний ресурс] / Code in Seoul // Medium. – Режим доступу: https://medium.com/@codeinseoul/the-complete-c-build-system-cmake-ninja-part-1-b7309be1a11e. – Дата звернення: 01.10.2025.
19. Cross Compiler CMake Usage Guide with rsynced Raspberry Pi 64 bit OS [Електронний ресурс] / GitHub. – Режим доступу: https://github.com/abhiTronix/raspberry-pi-cross-compilers/wiki/Cross-Compiler

-CMake-Usage-Guide-with-rsynced-Raspberry-Pi-64-bit-OS. – Дата звернення: 01.10.2025.

20. npm-start [Електронний ресурс] / npm Docs. – Режим доступу: https://docs.npmjs.com/cli/v9/commands/npm-start. – Дата звернення: 05.10.2025.
21. Svelte [Електронний ресурс] / WebDriverIO. – Режим доступу: https://webdriver.io/uk/docs/component-testing/svelte/. – Дата звернення: 11.10.2025.

ДОДАТОК А

Лістинг файлу rpi-toolchain.cmake

```
set(CMAKE_SYSTEM_NAME Linux)
set(CMAKE_SYSTEM_PROCESSOR aarch64)
set(CMAKE_C_COMPILER aarch64-linux-gnu-gcc)
set(CMAKE_CXX_COMPILER aarch64-linux-gnu-g++)
set(SYSROOT /home/USER/rpi-sysroot)
set(CMAKE_SYSROOT ${SYSROOT})
set(CMAKE_FIND_ROOT_PATH ${SYSROOT})
set(CMAKE_FIND_ROOT_PATH_MODE_PROGRAM NEVER)
set(CMAKE_FIND_ROOT_PATH_MODE_LIBRARY ONLY)
set(CMAKE_FIND_ROOT_PATH_MODE_INCLUDE ONLY)
```

ДОДАТОК Б

CMakeLists.txt для побудови серверної програми

```
cmake_minimum_required(VERSION 3.14)
project(ServerApp)

# Встановлення стандарту C++
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED ON)
set(CMAKE_AUTOMOC ON)

# JSON бібліотека (якщо це nlohmann/json.hpp)
include_directories(include)
# Підключаємо Crow
add_subdirectory(third_party/crow)
# Знаходимо необхідні пакети
find_package(Qt6 REQUIRED COMPONENTS Core Gui Widgets Qml Quick
QuickControls2 Sql Multimedia Pdf PdfWidgets HttpServer WebSockets)
find_package(Boost REQUIRED COMPONENTS log filesystem system)

# Збір усіх .cpp файлів
set(SOURCES
   main.cpp
   core/DatabaseManager.cpp
   providers/rest_api.cpp
   providers/SongProvider.cpp
   providers/UserProvider.cpp
   providers/ProgramProvider.cpp
   providers/WSServer.cpp
   providers/VideoProvider.cpp
   providers/SlideProvider.cpp
   providers/AudioProvider.cpp
   core/ContentItem.cpp
   models/Song.cpp
   models/User.cpp
   models/Program.cpp
)
# Додати виконуваний файл
add_executable(ServerApp ${SOURCES})

# Підключаємо залежності
target_link_libraries(ServerApp PRIVATE
```

```
    Qt6::Core
    Qt6::Gui
    Qt6::Widgets
    Qt6::Qml
    Qt6::Quick
    Qt6::QuickControls2
    Qt6::Sql
    Qt6::Multimedia
    Qt6::HttpServer
    Qt6::WebSockets
    Boost::log
    Boost::filesystem
    Boost::system
    Crow::Crow
)
target_include_directories(ServerApp PRIVATE
    ${PROJECT_SOURCE_DIR}
    ${PROJECT_SOURCE_DIR}/models
    ${PROJECT_SOURCE_DIR}/core
    ${PROJECT_SOURCE_DIR}/providers
    ${PROJECT_SOURCE_DIR}/include
)
# Встановлюємо дефіни
target_compile_definitions(ServerApp PRIVATE
    QT_DISABLE_DEPRECATED_BEFORE=0x060000
    APP_VERSION="${PROJECT_VERSION}"
)
```

ДОДАТОК В

Файл CMakeLists.txt для побудови десктопної програми

```
cmake_minimum_required(VERSION 3.16)
project(MSProjector VERSION 1.0 LANGUAGES CXX)
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED ON)
set(CMAKE_AUTOMOC ON)
set(CMAKE_AUTORCC ON)
set(CMAKE_AUTOUIC ON)

# Встановлення шляхів для вихідних файлів
set(CMAKE_RUNTIME_OUTPUT_DIRECTORY ${CMAKE_BINARY_DIR}/bin)
set(CMAKE_LIBRARY_OUTPUT_DIRECTORY ${CMAKE_BINARY_DIR}/lib)

# Знаходимо Qt пакети
find_package(Qt6 COMPONENTS Core Gui Widgets Qml Quick
QuickControls2 Sql Multimedia Pdf PdfWidgets WebEngineQuick
WebEngineCore Svg WebSockets Network REQUIRED)
# Додаємо підкаталог src
add_subdirectory(src)

# Встановлення цілей
install(TARGETS MSProjector
   BUNDLE DESTINATION .
   LIBRARY DESTINATION ${CMAKE_INSTALL_LIBDIR}
   RUNTIME DESTINATION ${CMAKE_INSTALL_BINDIR}
)
# Пакування
set(CPACK_GENERATOR "DragNDrop")
set(CPACK_PACKAGE_NAME "MSProjector")
set(CPACK_PACKAGE_VENDOR "MSProjector Team")
set(CPACK_PACKAGE_DESCRIPTION_SUMMARY "MSProjector - система для
дистанційного керування виводом на екран")
set(CPACK_PACKAGE_VERSION ${PROJECT_VERSION})
set(CPACK_PACKAGE_VERSION_MAJOR ${PROJECT_VERSION_MAJOR})
set(CPACK_PACKAGE_VERSION_MINOR ${PROJECT_VERSION_MINOR})
set(CPACK_PACKAGE_VERSION_PATCH ${PROJECT_VERSION_PATCH})
set(CPACK_PACKAGE_INSTALL_DIRECTORY "MSProjector")
set(CPACK_PACKAGE_ICON
"${CMAKE_SOURCE_DIR}/src/icons/msprojector.icns")
include(CPack)
```

CMakeLists.txt в папці src

```
# Збираємо всі файли джерел рекурсивно
file(GLOB_RECURSE SOURCES
    "*.cpp"
    "*.h"
)
# Ресурси QML та інші ресурси
set(RESOURCES
    ${CMAKE_CURRENT_SOURCE_DIR}/resources.qrc
)

# Список вихідних файлів
set(SOURCES
    main.cpp
    resources.qrc
    ui/MainWindow.cpp
    core/ApiConnection.cpp
    core/Application.cpp
    core/DatabaseManager.cpp
    core/PostgresDatabaseConnection.cpp
    core/SettingsManager.cpp
    core/ContentItem.cpp
    core/SearchService.cpp
    core/ServiceManager.cpp
    core/ProgramItemCache.cpp
    core/WebSocketConnection.cpp
    core/UploadProgressManager.cpp
    core/SlideUploadService.cpp
    core/WSServer.cpp
    providers/IContentProvider.cpp
    providers/SongProvider.cpp
    providers/SlideProvider.cpp
    providers/ProgramProvider.cpp
    models/Song.cpp
    models/Slide.cpp
    models/Program.cpp
    controllers/DisplayController.cpp
    core/UploadProgressManager.cpp
)
# Створюємо цільовий виконуваний файл
add_executable(MSProjector
    ${SOURCES}
```

```
    ${RESOURCES}
)
# Групуємо файли за директоріями в IDE
source_group(TREE "${CMAKE_CURRENT_SOURCE_DIR}" FILES ${SOURCES})
# Підключаємо залежності Qt
target_link_libraries(MSProjector PRIVATE
    Qt6::Core
    Qt6::Gui
    Qt6::Widgets
    Qt6::Qml
    Qt6::Quick
    Qt6::QuickControls2
    Qt6::Sql
    Qt6::Multimedia
    Qt6::Svg
    Qt6::Network
    Qt6::WebSockets
    Qt6::WebEngineQuick
    Qt6::WebEngineCore
)
# Встановлюємо шляхи включення
target_include_directories(MSProjector PRIVATE
    ${CMAKE_CURRENT_SOURCE_DIR}
    ${CMAKE_CURRENT_SOURCE_DIR}/core
    ${CMAKE_CURRENT_SOURCE_DIR}/providers
    ${CMAKE_CURRENT_SOURCE_DIR}/models
    ${CMAKE_CURRENT_SOURCE_DIR}/controllers
    ${CMAKE_CURRENT_SOURCE_DIR}/ui
    ${CMAKE_CURRENT_SOURCE_DIR}/logging
    ${CMAKE_CURRENT_SOURCE_DIR}/qml
)
# Встановлюємо дефіни
target_compile_definitions(MSProjector PRIVATE
    QT_DISABLE_DEPRECATED_BEFORE=0x060000
    APP_VERSION="${PROJECT_VERSION}"
)
```