

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ
ВІДТВОРЕННЯ АУДІОФАЙЛІВ
SOFTWARE FOR AN INFORMATION SYSTEM OF
AUDIO PLAYBACK

Виконав(ла): студент(ка) 4 курсу, групи КНТ-211
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерні науки

УДОД А.А.

(ПРИЗВИЩЕ та ініціали)

Керівник ГОФМАН Є.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент СКРУПСЬКИЙ С.Ю.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)

Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

УДОД Аліси Анатоліївни

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення інформаційної системи відтворення аудіофайлів. Software for an Information System of Audio Playback

керівник проєкту (роботи) к.т.н., ГОФМАН Євгеній Олександрович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 03 червня 2025 року

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

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

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

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ГОФМАН Є.О., доцент		
Нормоконтроль	КАМІНСЬКА Ж.К., асистент		

7. Дата видачі завдання “ 28 ” квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

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

Студент(ка)

(підпис) Аліса УДОД
(Імя ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Євгеній ГОФМАН
(Імя ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
88 с., 3 табл., 27 рис., 3 дод., 18 джерел.

C#, REACT, MONODEVELOP, МОВА ПРОГРАМУВАННЯ,
ПРОГРАМНІ ЗАСОБИ.

Об'єкт дослідження – процес розробки засобів для відтворення аудіофайлів.

Предмет дослідження – засоби для відтворення аудіофайлів.

Мета роботи – розробка програмного забезпечення для відтворення аудіофайлів.

Матеріали, методи та технічні засоби: мова програмування C#, середовище розробки MonoDevelop.

Результати. Розроблено проєктні рішення для створення програмного забезпечення для відтворення аудіофайлів. Створено програмне забезпечення інформаційної системи для відтворення аудіофайлів. Проведено тестування розробленого програмного забезпечення для відтворення аудіофайлів.

Висновки. Мету роботи досягнуто. Розроблено програмне забезпечення для відтворення аудіофайлів за допомогою мови програмування C# та середовища розробки MonoDevelop.

Галузь використання – відтворення аудіофайлів.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 88 pages, 3 tables, 27 figures, 3 appendixes, 18 sources.

C#, REACT, MONODEVELOP, PROGRAMMING LANGUAGE, SOFTWARE.

The object of research is the process of developing tools for playing audio files.

The subject of the research is tools for playing audio files.

The purpose of this work is to develop the software for playing audio files.

Materials, methods and technical tools: C# programming language, MonoDevelop development environment.

Results. Project solutions for the creation of software for playing audio files have been designed. The software for playing audio files has been developed. The developed software for playing audio files has been tested.

Conclusions. The goal of the work has been achieved. The software tools for playing audio files using the C# programming language and the MonoDevelop development environment have been developed.

Scope of use – playing audio files.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ	9
1 Аналіз предметної області	11
1.1 Інформаційні системи для відтворення аудіофайлів	11
1.2 Аналіз інформаційних систем для відтворення аудіофайлів.....	20
1.3 Висновки за розділом 1	29
2 Матеріали і методи	31
2.1 Вибір мови програмування.....	31
2.2 Вибір середовища розробки для створення програмного забезпечення інформаційної системи для відтворення аудіофайлів.....	34
2.3 Висновки за розділом 2.....	37
3 Опис програми	38
3.1 Структура програмного забезпечення інформаційної системи для відтворення аудіофайлів	38
3.2 Функціонування програмного забезпечення інформаційної системи для відтворення аудіофайлів	40
3.3 Розробка діаграми прецедентів програмного забезпечення інформаційної системи для відтворення аудіофайлів.....	43
3.4 Проектування інтерфейсу програмного забезпечення інформаційної системи для відтворення аудіофайлів.....	44
3.5 Висновки за розділом 3.....	47
4 Експлуатація, тестування та експериментальне дослідження програми.....	48
4.1 Призначення й умови застосування програми	48
4.2 Характеристики програмного забезпечення інформаційної системи для відтворення аудіофайлів	49
4.3 Інструкція по експлуатації програми.....	51
4.3.1 Звернення до програми.....	51
4.3.2 Вхідні й вихідні дані.....	51

4.3.3 Повідомлення.....	51
4.4 Виконання програмного забезпечення інформаційної системи для відтворення аудіофайлів.....	51
4.5 Тестування програмного забезпечення інформаційної системи для відтворення аудіофайлів.....	55
4.6 Висновки за розділом 4.....	56
Висновки.....	57
Перелік джерел посилань.....	60
Додаток А Технічне завдання.....	62
Додаток Б Фрагмент тексту програми	66
Додаток В Слайди презентації	81

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

БД – база даних;

ІС – інформаційна система;

ПЗ – програмне забезпечення;

ПК – персональних комп'ютерів.

ВСТУП

У сучасному цифровому суспільстві аудіоінформація є невід'ємною частиною повсякденного життя, відповідно, з розвитком інформаційних технологій зростає потреба в ефективному та зручному програмному забезпеченні для відтворення аудіофайлів. Такі програмні системи забезпечують високу якість звуку, підтримку широкого спектру аудіоформатів, інтеграцію з іншими сервісами (наприклад, хмарними сховищами, онлайн бібліотеками), а також мають зручний інтерфейс користувача [1], [2].

Зростання обсягів мультимедійного контенту, поширення потокових сервісів, популяризація підкастів та аудіокниг формує високий попит на інноваційні програмні рішення, здатні адаптуватися до різних платформ, зокрема, мобільних пристроїв та персональних комп'ютерів (ПК). Актуальність розробки програмного забезпечення (ПЗ) для відтворення аудіофайлів також посилюється необхідністю доступу до аудіоконтенту людьми з особливими потребами, для яких такі системи можуть відігравати важливу роль у соціалізації та навчанні [3], [4].

Крім того, в умовах глобалізації та інформаційного перенасичення важливим є створення програм, які дозволяють швидко та ефективно організовувати, відтворювати й керувати великими колекціями аудіофайлів. Це відкриває перспективи для впровадження елементів штучного інтелекту, персоналізованих рекомендацій, голосового керування тощо [5]-[7].

Проте деякі програмні засоби для відтворення аудіофайлів мають ряд недоліків, які можуть впливати на їх ефективність та зручність використання. Одним із головних проблем є обмежена підтримка форматів, адже не всі програми здатні працювати з усіма популярними або спеціалізованими форматами аудіофайлів. Це може вимагати додаткових конвертацій або встановлення спеціальних плагінів, що значно ускладнює процес використання. Ще одним важливим недоліком є високі системні вимоги, які

можуть призводити до погіршення продуктивності інших застосунків або навіть самого пристрою, особливо на старих чи малопотужних пристроях. Багато програм також мають надто простий інтерфейс, що не завжди підходить користувачам, яким важливі додаткові налаштування та можливість персоналізації програми. Деякі програми також не надають достатньої якості звуку, особливо на дешевих або старих пристроях, через погану оптимізацію або стиснення звукових файлів, що може негативно вплинути на досвід користувача.

Тому актуальною є розробка програмного забезпечення інформаційної системи відтворення аудіофайлів. У дипломній кваліфікаційній роботі бакалавра розв'язується актуальне завдання розробки програмного забезпечення для відтворення аудіофайлів.

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

- виконати аналіз предметної області та програмного забезпечення для відтворення аудіофайлів;
- здійснити проєктування програмного забезпечення для відтворення аудіофайлів;
- створити програмного забезпечення для відтворення аудіофайлів;
- виконати тестування розробленого програмного забезпечення інформаційної системи для відтворення аудіофайлів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Інформаційні системи для відтворення аудіофайлів

Сфера розробки рішень для прослуховування музики постійно розвивається, і конкуренція в цьому напрямку досить висока. Програмні продукти, здатні відтворювати широкий спектр аудіоформатів, працювати безперервно та надавати якісний звук, мають перевагу в очах користувачів. Особливо цінується можливість налаштовувати інтерфейс, адаптувати функціональність і підключати додаткові модулі чи сервіси [1].

Програмне забезпечення для відтворення аудіофайлів призначено для прослуховування музики чи будь-яких інших звукових записів у цифровому форматі. Таке ПЗ забезпечує підтримку різноманітних аудіоформатів, дозволяє управляти списками відтворення, налаштовувати якість звучання, а також може мати функції потокового передавання або синхронізації з іншими сервісами [1].

Серед головних характеристик програм для відтворення аудіо — зручність інтерфейсу, швидкість обробки файлів, стабільна робота та можливість персоналізації. Деякі програми зосереджені на мінімалізмі й простоті, інші ж — на широкій функціональності та розширюваності через плагіни або модулі. Таке програмне забезпечення може бути локальним або мати функцію доступу до онлайн бібліотек і сервісів. Усі ці особливості дозволяють користувачеві слухати улюблену музику з максимальною зручністю й якістю [1], [2].

Розроблення застосунку для прослуховування музики передбачає проходження послідовного процесу, що включає підготовчі дії та стратегічне планування. Все починається з ретельного вивчення майбутнього продукту: аналізується ситуація на ринку, виявляються потреби потенційних користувачів і формулюється чітке бачення очікуваного результату. Під час початкового етапу уточнюються ідеї, визначаються технічні параметри, складаються вимоги та окреслюється концепція проєкту [1], [2].

На основі зібраних даних обирається тип програмного рішення. Можна зосередитися на сервісі з доступом до медіатеки, що дає змогу слухати музику онлайн, або ж передбачити збереження особистих аудіофайлів у хмарі для подальшого керування ними. Існує й можливість реалізувати платформу з потоковим радіо, яка відкриває доступ до ефірів з усього світу. Найчастіше обирається варіант з доступом до централізованої музичної бібліотеки, адже він забезпечує гнучкість і широкий вибір контенту для слухачів [1], [2].

Після визначення формату програмного забезпечення для відтворення аудіофайлів плануються його ключові можливості. На цьому етапі увага зосереджується на тому, як забезпечити зручність, надійність та ефективність взаємодії користувача з додатком [1], [2].

Під час створення програмного забезпечення для відтворення аудіофайлів особливу увагу приділяють функціональним можливостям, адже саме вони формують враження користувача та визначають подальший інтерес до програмного продукту. Щоб забезпечити якісне сприймання музичного контенту, передбачають різноманітні інструменти, які полегшують пошук, відкриття та організацію звуку [1], [2].

Особливе значення надається механізмам, що сприяють виявленню нових композицій. Завдяки аналітичним підходам і сучасним технологіям обробки даних, користувачі отримують можливість прослуховувати треки, які відповідають їхнім попереднім уподобанням. Рекомендаційні системи автоматично визначають релевантний контент, ґрунтуючись на історії взаємодії, стилістичних параметрах композицій і текстових характеристиках [2], [3].

Окрему роль відіграють тематичні добірки та популярні хіти, які забезпечують орієнтацію в актуальному музичному середовищі. Завдяки цьому користувачі можуть знайомитися з найпоширенішими аудіозаписами певного періоду або географічного регіону. Такі добірки не тільки збагачують досвід слухача, а й стимулюють інтерес до нових виконавців [2], [3].

Щоб зробити взаємодію ще зручнішою, у програмному забезпеченні реалізуються можливості створення й керування списками композицій, які формуються за жанровими ознаками, емоційними станами або призначенням. Такі добірки використовуються як фоновий супровід для різних занять чи ситуацій [2], [3].

Крім цього, забезпечується швидкий доступ до конкретного контенту за допомогою інтегрованого пошуку. Пошукові механізми в таких програмах працюють не лише за назвами, а й розпізнають текстові фрагменти, що дає змогу знаходити музику навіть за окремими рядками [2], [3].

Інтерфейс програмного забезпечення для відтворення аудіофайлів також передбачає інформаційні блоки, присвячені окремим виконавцям. У таких розділах зібрано матеріали, які допомагають ознайомитися з творчістю артистів: дискографією, новими треками, добірками та найпопулярнішими композиціями. Це сприяє глибшому залученню слухачів і формуванню індивідуального музичного середовища [2], [3].

У програмному забезпеченні, призначеному для прослуховування музики, реалізується низка можливостей, що забезпечують комфортне керування аудіофайлами. Основна мета такого застосунку полягає у швидкому доступі до відтворення треків, їх навігації та безперервному звучанні, навіть за умов обмеженого або нестабільного підключення до мережі. Важливо, щоб інтерфейс залишався інтуїтивно зрозумілим, а відгук системи – миттєвим, що дає змогу уникати затримок під час взаємодії з аудіо [2], [3].

У застосунках цього типу передбачено низку стандартних елементів керування, що дозволяють перемикатися між піснями, зупиняти або запускати трек, налаштовувати чергу відтворення, а також використовувати опції повтору чи випадкового порядку. Часто додається можливість ознайомлення з текстом композиції без необхідності звертатися до сторонніх джерел, що особливо корисно під час вивчення іноземних мов або уважного прослуховування змісту пісні [2], [3].

Функція обміну музичним вмістом також відіграє помітну роль. Передбачено надсилання улюблених треків іншим користувачам, що, водночас, сприяє поширенню самого додатку. Додаткові дії включають взаємодію з бібліотекою – додавання, збереження, впорядкування або видалення треків, а також можливість миттєвого переходу до сторінки виконавця чи альбому [2], [3].

У деяких випадках користувачі можуть залишатися без доступу до мережі, тому можливість відтворення офлайн залишається вкрай актуальною. Завдяки попередньому збереженню музики, відтворення може здійснюватися без підключення до Інтернету, що забезпечує зручність у дорозі або за умов нестабільного сигналу. Часто така функціональність доступна в рамках підписки, що водночас стимулює зацікавлення платними можливостями сервісу [2], [3].

Після авторизації відкривається доступ до персоналізованого простору. У ньому зручно налаштовувати добірки, змінювати черговість треків, переглядати улюблені пісні чи створювати нові списки для прослуховування. Наявність гнучкого керування бібліотекою сприяє індивідуалізації взаємодії з музичним контентом [2], [3].

Для розширення функціональності програмного забезпечення для відтворення аудіофайлів реалізуються додаткові можливості: повідомлення з оновленнями щодо улюблених виконавців, підтримка розпізнавання треків за уривками тексту або звучанням, взаємодія з голосовими командами для зручного управління, а також підтримка створення спільних добірок з іншими користувачами. Окремо у програмному забезпеченні для відтворення аудіофайлів можна передбачити інтеграцію з платформами для придбання квитків на концерти, що відкриває додаткові шляхи монетизації [2], [3].

Після проведення всебічного аналізу та визначення основної концепції застосунку для відтворення аудіофайлів, переходять до етапу оформлення зовнішнього вигляду та реалізації логіки інтерфейсу. На цьому рівні визначаються основні візуальні та функціональні принципи, які мають

забезпечити зручність користування, простоту взаємодії та стабільність роботи на різних пристроях [3], [4].

У процесі формування візуального середовища особлива увага приділяється гармонійному поєднанню естетики та функціональності. Вважається важливим уникати перевантаження інтерфейсу зайвими елементами, натомість фокусуючись на легкості навігації та швидкому доступі до основних функцій. Платформи для прослуховування музики, які вже здобули популярність, демонструють доцільність мінімалістичних рішень, де головну роль відіграє користувацький досвід, а не візуальні ефекти [3], [4].

Підготовка до розробки програмного забезпечення для відтворення аудіофайлів передбачає створення концептуального плану, в якому розміщення функціональних блоків та елементів управління визначається відповідно до зручності користувача. На ранніх етапах використовуються умовні ескізи без кольору й графіки, що допомагає зосередитися на загальній структурі. Пізніше готуються візуальні шаблони з урахуванням стилістики, шрифтів, кольорової палітри та елементів бренду. Завершальним кроком цього циклу стає створення інтерактивної моделі, яка дозволяє протестувати навігацію та сценарії користувача ще до початку написання коду [3], [4].

Коли основа візуального інтерфейсу програмного забезпечення для відтворення аудіофайлів готова, переходять до реалізації функціональної частини. Розробка здійснюється відповідно до технічного завдання з урахуванням особливостей обраної платформи, характеристик пристроїв, специфікацій операційних систем та вимог до продуктивності. Залежно від складності задуму, можуть використовуватись різні інструменти та технологічні рішення, включно з мовами програмування, фреймворками та бібліотеками. Для забезпечення стабільної роботи застосунку важливою є компетентність розробників у цих технологіях [3], [4].

Фінансовий аспект розробки програмного забезпечення для відтворення аудіофайлів залежить від великої кількості чинників: обсягу функціоналу, кількості платформ, що підтримуються, обраних технологій,

тривалості розробки та рівня складності. Без аналізу конкретного проєкту надати точну оцінку складно, однак загальні тенденції дають змогу окреслити приблизні рамки вартості. Чим складніший застосунок і більший набір його можливостей, тим більше часу та ресурсів буде потрібно для його повноцінного запуску [3], [4].

Перед випуском програмного забезпечення, призначеного для відтворення аудіофайлів, важливо провести ретельну перевірку його функціонування. Паралельно з розробкою виконується оцінка стабільності роботи інтерфейсу та логіки програми. Весь функціонал перевіряється з точки зору відповідності очікуваній поведінці, а також проводиться пошук і документування можливих помилок [3], [4].

Суть перевірки полягає в аналізі роботи окремих модулів програмного забезпечення для відтворення аудіофайлів, оцінці зручності використання та реакції системи на різні дії користувача. Крім цього, перевіряється взаємодія з різними типами пристроїв і платформ, зокрема мобільними, настільними та вебрішеннями. Особливу увагу приділяють швидкодії, чутливості інтерфейсу, здатності до масштабування та захисту персональних даних. Безпека має критичне значення, тому також досліджуються способи автентифікації, збереження даних і запобігання зовнішнім втручанням [3], [4].

У процесі впровадження змін до програмного забезпечення для відтворення аудіофайлів важливо пересвідчитись, що нові елементи не порушують уже реалізовану функціональність. Уникнення технічних збоїв, візуальних дефектів або повільного реагування забезпечує більш позитивне сприйняття серед користувачів [4], [5].

Однак технічна надійність не єдине, що слід враховувати. Для програмного забезпечення, яке дозволяє відтворювати музику або інший аудіоконтент, важливо дотримуватись правових норм. Щоб уникнути юридичних проблем, необхідно заздалегідь узгодити всі умови використання контенту. Всі аудіофайли, які передаються або зберігаються у застосунку,

повинні мати відповідні дозволи, які регулюють їхнє відтворення, розповсюдження й публічне звучання [4], [5].

Залежно від регіону та специфіки програмного забезпечення для відтворення аудіофайлів, можуть знадобитися дозволи на використання оригінальних записів, права на музику як твір, а також дозвіл на її трансляцію. Усі ці аспекти визначаються за участі правовласників або організацій, які займаються управлінням авторських прав. Без оформлення таких угод функціонування програми в правовому полі є неможливим, і це може призвести до юридичної відповідальності та фінансових санкцій. Тому перед запуском застосунку необхідно переконатись, що всі ліцензії оформлено коректно [4], [5].

Після розгляду ключових технічних етапів створення платформи для прослуховування музики, варто зосередити увагу на механізмах отримання прибутку. У сфері програмного забезпечення, що забезпечує відтворення аудіоконтенту, джерелами доходу часто стають платні доступи та використання рекламного контенту [4], [5].

Найбільш поширеним підходом є модель з обмеженим функціоналом для загального доступу, де розширені можливості доступні лише після оплати. У таких випадках забезпечується безкоштовне використання з базовими функціями, тоді як додаткові сервіси, наприклад, прослуховування треків без підключення до мережі, відкриваються лише підписникам преміумрівня. Успішне застосування цієї стратегії можливе завдяки ретельному балансу між привабливістю безкоштовного доступу та привабливістю розширених можливостей [4], [5].

Ще одним джерелом прибутку є інтеграція рекламних вставок у процес взаємодії з програмним забезпеченням для відтворення аудіофайлів. В аудіопросторі реклама зазвичай вбудовується в потік відтворення та з'являється між композиціями. У випадку зростання аудиторії платформи, така модель дозволяє забезпечити стабільне фінансування без необхідності вносити зміни до базової функціональності продукту [4], [5].

Створення конкурентоспроможного аудіосервісу не може обмежуватися лише реалізацією стандартного набору функцій. Конкуренція в цьому секторі вимагає інноваційного підходу, адаптивних рекомендаційних алгоритмів, привабливого інтерфейсу та технічної надійності. Без цих компонентів важко досягти лояльності користувачів і зайняти стабільну позицію на ринку [5], [6].

Для досягнення належної якості програмного забезпечення для відтворення аудіофайлів необхідно співпрацювати з командою, яка має значний досвід в галузі цифрових рішень для розваг. Такий підхід дозволяє уникнути типових помилок, скоротити час на реалізацію і гарантувати, що розроблене рішення відповідатиме високим очікуванням аудиторії [5], [6].

Програмне забезпечення для відтворення музичних файлів продовжує залишатися актуальним навіть у добу стрімінгових сервісів, хоча його популярність значно зменшилася порівняно з періодом, коли більшість користувачів слухала музику безпосередньо з жорсткого диска. Незважаючи на поширення Інтернет платформ, які надають доступ до аудіоконтенту в режимі онлайн, усе ще існує потреба у локальному програвачі, здатному обробляти особисту музичну колекцію [5], [6].

Коли використовується такий застосунок, зазвичай на початку задається шлях до директорії, де зберігаються аудіофайли. Програма для відтворення аудіофайлів автоматично здійснює перегляд вмісту цієї теки, зчитує інформацію з кожного треку та формує зібрану бібліотеку. Цей процес може тривати певний час, особливо якщо програмне забезпечення використовує зовнішні бібліотеки для роботи з метаданими. Після початкового сканування наступні відкриття відбуваються швидше, адже результати зберігаються в тимчасових файлах [6], [7].

Ефективність таких програм залежить не лише від мови програмування, на якій їх реалізовано, але й від загального підходу до роботи з ресурсами системи. Часто застосовуються спеціальні техніки керування пам'яттю, що дозволяє зменшити навантаження на оперативну пам'ять під час

простою або активного відтворення. Особливо важливою є здатність програми адаптуватися до слабших пристроїв і не створювати зайвого навантаження, навіть коли одночасно обробляється велика кількість аудіофайлів [6], [7].

У деяких рішеннях основний акцент робиться на легкості інтерфейсу та мінімалізмі, тоді як інші прагнуть надати максимально багатofункціональний набір можливостей. Незалежно від підходу, головною метою залишається швидке завантаження треків, точне визначення даних про композиції та стабільна робота під час тривалого використання. І хоча оптимізація продуктивності залишається складним завданням, вона є ключовою умовою для забезпечення комфортного прослуховування музики з локального сховища [6], [7].

Щоб у мобільному застосунку була реалізована функція відтворення звукових файлів, використовуються інструменти, які дають змогу інтегрувати аудіо програвач без необхідності глибокого занурення в код. Розробка такого компонента здійснюється за допомогою візуального середовища, яке дозволяє створювати інтерфейс і логіку керування аудіо за допомогою зручного редактора [6], [7].

На початковому етапі розробки програмного забезпечення для відтворення аудіофайлів передбачається наявність середовища розробки з необхідними інструментами. В інтерфейсі проєкту додається окремий екран, призначений для керування звуковим контентом. У межах цього екрану розміщуються елементи керування – кнопки для запуску, зупинки й призупинення відтворення [6], [7].

Для забезпечення технічної можливості програвання звуку до проєкту розробки програмного забезпечення для відтворення аудіофайлів підключається спеціальний програмний модуль, який відповідає за роботу з аудіофайлами. Його підключення відбувається шляхом внесення відповідного запису в конфігураційний файл застосунку. Після цього програвач стає доступним для використання в кодовій частині [6], [7].

Щоб реалізувати логіку роботи елементів керування, оголошується об'єкт, що представляє сам програвач. До цього об'єкта додаються функції, які реагують на взаємодію користувача з кнопками: відтворення активує запуск аудіо з вказаного джерела, призупинення зупиняє відтворення тимчасово, а зупинка завершує його повністю [6], [7].

Коли обробники подій програмного забезпечення для відтворення аудіо файлів налаштовані, залишається поєднати ці функції з відповідними компонентами інтерфейсу. Це здійснюється через панель налаштувань, де для кожної дії вказується потрібна функція [6], [7].

Щоб переконатися у правильності реалізації, програмне забезпечення для відтворення аудіо файлів запускається в режимі попереднього перегляду або тестування. Після цього перевіряється робота всіх основних функцій аудіо програвача. За бажанням можливий подальший розвиток функціональності: відображення часу відтворення, налаштування гучності або підтримка кількох форматів аудіо. Така основа дає змогу розширювати можливості програвача залежно від потреб майбутнього застосунку [6], [7].

Визначено, що програмне забезпечення для відтворення аудіофайлів призначено для прослуховування музики чи будь-яких інших звукових записів у цифровому форматі. Таке ПЗ забезпечує підтримку різноманітних аудіоформатів, дозволяє управляти списками відтворення, налаштовувати якість звучання, а також може мати функції потокового передавання або синхронізації з іншими сервісами.

1.2 Аналіз інформаційних систем для відтворення аудіофайлів

Проаналізуємо інформаційні системи для відтворення аудіофайлів [8]-[14].

Програмне забезпечення Doramine (рис. 1.1) призначено для відтворення аудіофайлів і надає користувачеві зручне середовище для прослуховування музики. Основна мета цього застосунку — забезпечити

максимально просте і комфортне аудіовідтворення без ускладнених функцій чи перевантаженого інтерфейсу. У роботі з музичними файлами цей програвач вирізняється зрозумілим та привабливим оформленням, яке дозволяє легко орієнтуватися в налаштуваннях і функціоналі, навіть якщо користувач не має спеціальної технічної підготовки [8], [10], [12].

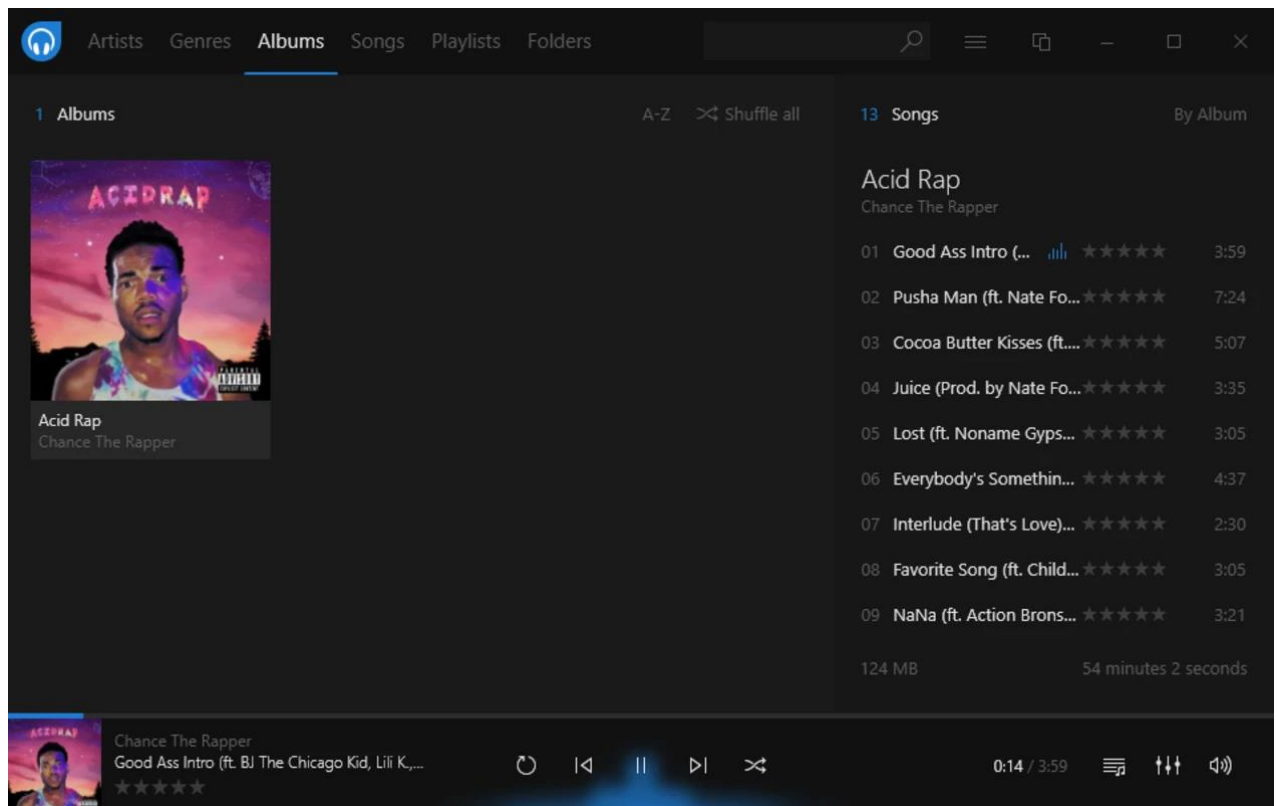


Рисунок 1.1 – Програмне забезпечення Doramine [12]

Відтворення аудіофайлів у Doramine реалізовано з акцентом на зручність та естетичне сприйняття. Інтерфейс орієнтовано на швидкий доступ до основних функцій, пов'язаних із прослуховуванням музики. Програма Doramine дає змогу не лише запускати аудіофайли, а й впорядковувати їх, працювати з тегами та створювати списки композицій, що полегшує керування особистою фонотекою [8], [10], [12].

Під час відтворення аудіофайлів можна використовувати вбудовані можливості еквалізації, щоб адаптувати звучання відповідно до власних уподобань. Підтримується велика кількість аудіоформатів, що забезпечує

сумісність із переважною більшістю музичних файлів. Doramine фокусується на тому, щоб процес відтворення був інтуїтивно зрозумілим і не вимагав жодних зайвих дій [8], [10], [12].

Програмне забезпечення Doramine розроблене як легкий інструмент для відтворення звуку, що дозволяє уникнути перевантаження системних ресурсів. Його використовують для щоденного прослуховування музики у комфортному середовищі без складних налаштувань. Усе це робить Doramine зручним рішенням для тих, хто цінує простоту і прагне якісного звучання при мінімальній кількості налаштувань [8], [10], [12].

Основними характеристиками програмного забезпечення Doramine є такі [8], [10], [12]:

- мінімалістичний інтерфейс: програмне забезпечення Doramine надає користувачам зручне і приємне візуальне оформлення без зайвих елементів інтерфейсу [8], [10], [12];

- підтримка аудіоформатів: програмне забезпечення Doramine дозволяє відтворювати музику у багатьох популярних форматах, включаючи MP3, WAV, FLAC та інші [8], [10], [12];

- простота використання: програмне забезпечення Doramine не потребує спеціальних знань щодо роботи зі звуковими файлами, підходить для будь-якої категорії користувачів [8], [10], [12];

- організація бібліотеки: програмне забезпечення Doramine дозволяє впорядковувати треки за виконавцями, альбомами, жанрами для зручної навігації [8], [10], [12];

- вбудований еквалайзер: програмне забезпечення Doramine надає можливість користувачам налаштовувати звучання відповідно до власних вподобань [8], [10], [12];

- редагування тегів: програмне забезпечення Doramine дає змогу змінювати метадані файлів для кращої каталогізації [8], [10], [12];

- легкість програми: програмне забезпечення Doramine споживає мінімум системних ресурсів і швидко завантажується [8], [10], [12];

– обмежені додаткові функції: програмне забезпечення Doramine не підтримує потокову музику чи складну персоналізацію [8], [10], [12].

Серед недоліків програмного забезпечення Doramine можна відзначити обмежений набір додаткових функцій, які могли б бути корисними для просунутих користувачів. Відсутність підтримки потокових сервісів, автоматичного завантаження обкладинок альбомів або вбудованої синхронізації з хмарними сховищами дещо знижує його функціональність порівняно з більш універсальними аудіоплеєрами. Крім того, можливості налаштування інтерфейсу залишаються досить вузькими, що може не задовольнити користувачів, які надають перевагу повній персоналізації зовнішнього вигляду програми [8], [10], [12].

Програмне забезпечення Clementine (рис. 1.2) використовується для відтворення аудіофайлів на різних операційних системах, зокрема Windows, Linux та Mac [9], [10], [13].

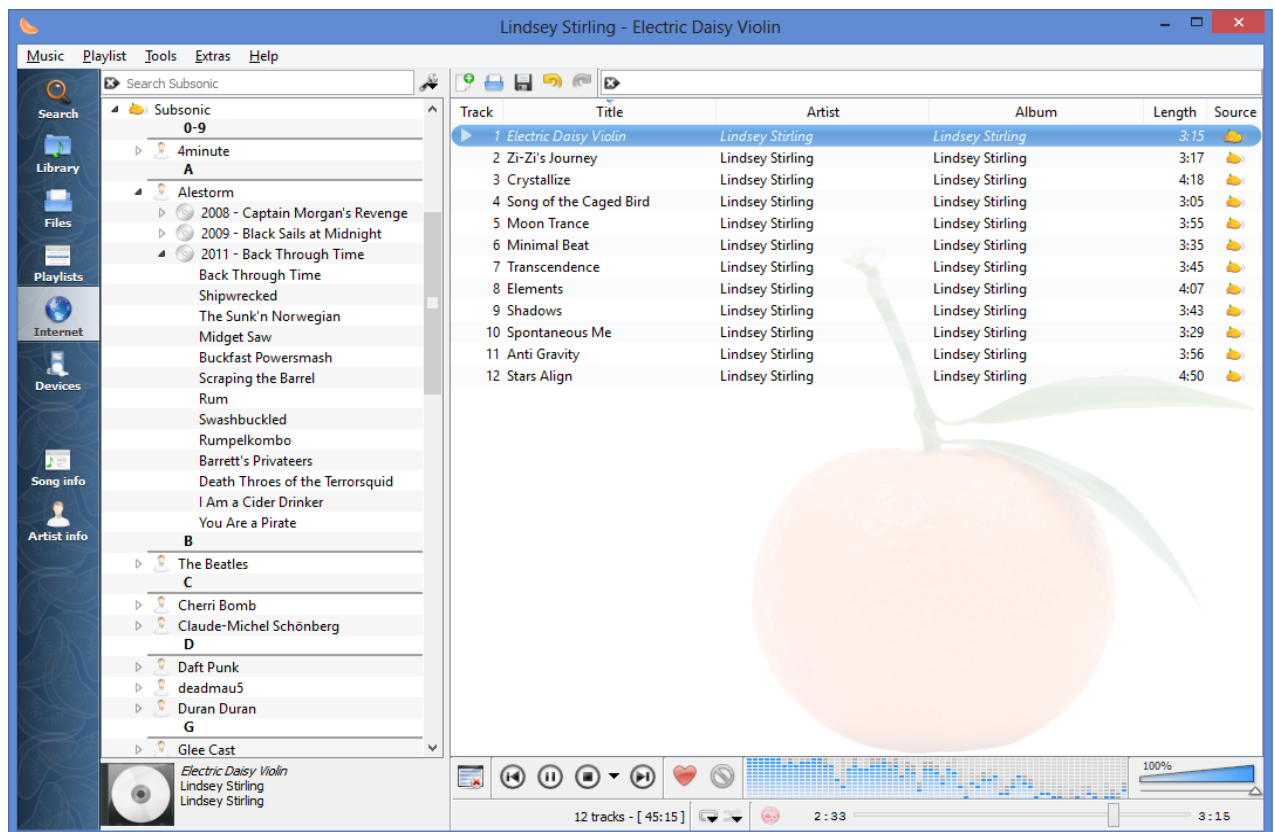


Рисунок 1.2 – Програмне забезпечення Clementine [13]

Програмне забезпечення Clementine часто обирається тими, хто шукає багатофункціональний аудіоплеєр для прослуховування музичних треків. Його функціональність дозволяє ефективно організовувати та програвати аудіофайли, створювати списки відтворення, користуватись вбудованим еквалайзером і бібліотекою музики. Програмне забезпечення Clementine надає зручне управління відтворенням аудіофайлів завдяки інтуїтивно зрозумілому інтерфейсу, хоча можливості налаштування зовнішнього вигляду є обмеженими [9], [10], [13].

Програмне забезпечення Clementine призначено не лише для локального відтворення аудіофайлів, а й для прослуховування Інтернет радіо. Крім того, підтримується редагування тегів і конвертація аудіоформатів, що дозволяє гнучко працювати з музичними файлами. Незалежно від того, яка платформа використовується, це програмне забезпечення дозволяє зберегти однаковий підхід до відтворення музики. Його стабільність та зручність роблять Clementine придатним для користувачів, які регулярно відтворюють аудіофайли і хочуть мати у своєму розпорядженні універсальний інструмент для цієї мети [9], [10], [13].

Програмне забезпечення Clementine має такі особливості [9], [10], [13]:

- кросплатформна підтримка: програмне забезпечення для відтворення аудіофайлів Clementine працює на операційних системах Windows, Linux та macOS без втрати функціональності [9], [10], [13];

- багатофункціональний інтерфейс: програмне забезпечення Clementine надає доступ до списків відтворення, музичної бібліотеки, еквалайзера та перегляду пісень для зручного керування відтворенням аудіофайлів користувачів [9], [10], [13];

- відтворення інтернет-радіо: програмне забезпечення Clementine підтримує трансляції музики через онлайн радіостанції, що розширює можливості прослуховування аудіофайлів [9], [10], [13];

– вбудовані інструменти: програмне забезпечення Clementine включає редактор тегів, конвертер аудіо та інші функції, що доповнюють основні можливості програми для відтворення аудіофайлів [9], [10], [13];

– обмежені можливості дизайну: інтерфейс програмного забезпечення Clementine має мінімальні налаштування зовнішнього вигляду, що може не відповідати очікуванням користувачів, які шукають візуальну гнучкість у програмах для відтворення аудіофайлів [9], [10], [13].

Серед недоліків програмного забезпечення Clementine можна відзначити обмежені можливості персоналізації інтерфейсу, що не дозволяє змінити зовнішній вигляд програвача відповідно до індивідуальних уподобань користувача. Також іноді виникають труднощі з оновленням або підтримкою нових музичних сервісів, що може впливати на комфорт при відтворенні аудіофайлів з онлайн джерел. Крім того, програма має досить консервативний дизайн, який може здатися застарілим для тих, хто віддає перевагу сучасному стилю у відтворенні аудіофайлів [9], [10], [13].

Програмне забезпечення Foobar (рис. 1.3) призначене для відтворення аудіофайлів, яке вирізняється високою функціональністю та можливістю тонкого налаштування. Основна мета цієї програми полягає в забезпеченні якісного й стабільного відтворення звуку незалежно від формату аудіофайлу. Завдяки гнучкій архітектурі, що дозволяє додавати додаткові компоненти, це програмне забезпечення дає змогу суттєво розширити набір інструментів для роботи з аудіофайлами. Саме через модульну структуру програма Foobar легко адаптується до найрізноманітніших вимог користувача щодо прослуховування музики. Особливу увагу в цьому програмному забезпеченні приділено підтримці великої кількості аудіоформатів, зокрема як зі стисненням, так і без втрат. Програмне забезпечення Foobar використовується для відтворення композицій, створення списків відтворення, управління бібліотекою музики, редагування тегів, копіювання аудіо з компакт-дисків та перетворення звукових файлів між форматами [8]-[11], [14].

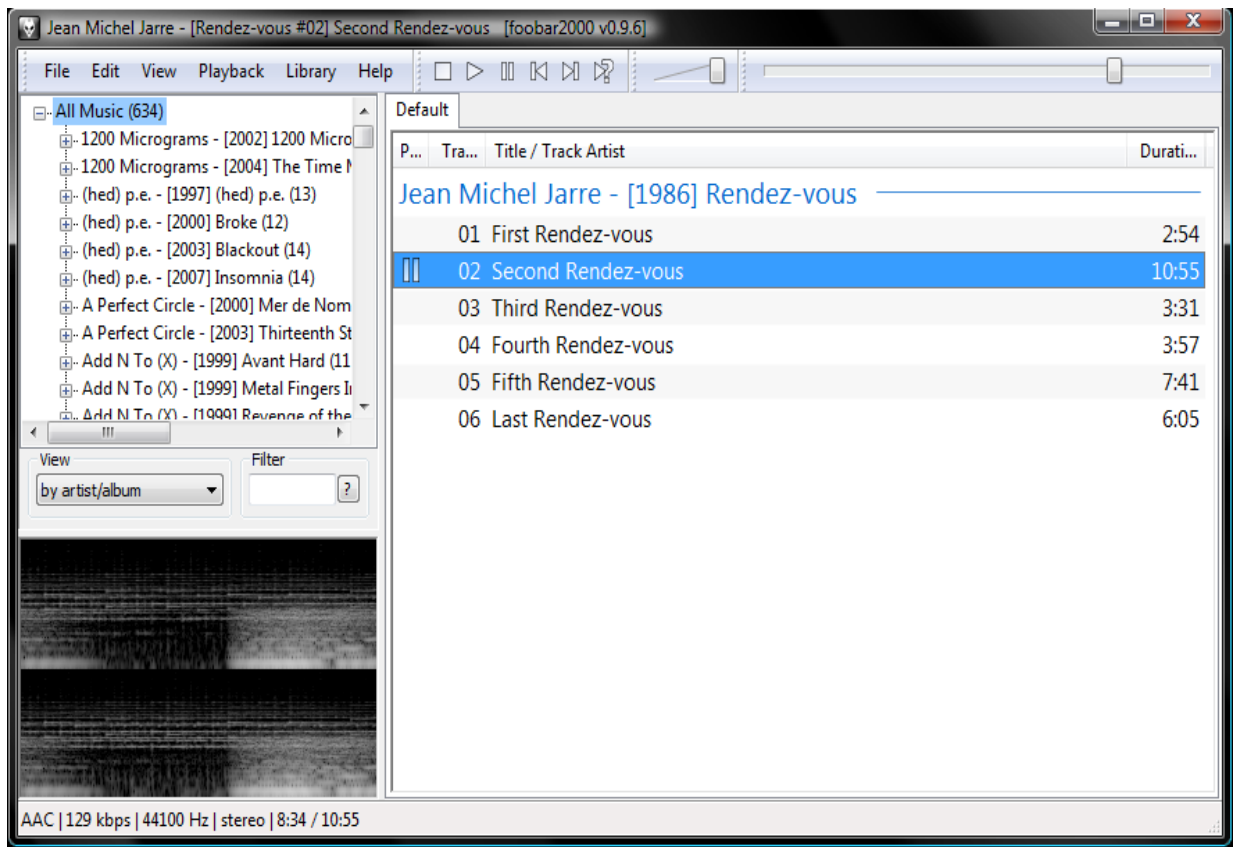


Рисунок 1.3 – Програмне забезпечення Foobar [14]

Інтерфейс у програмі Foobar реалізовано максимально просто, що забезпечує зосередженість на головному — точному й стабільному відтворенні аудіофайлів [8]-[11], [14].

Цей аудіоплеєр підтримує безперервне програвання, нормалізацію гучності та широкі можливості для гарячих клавіш, що дає змогу здійснювати повноцінне керування процесом відтворення аудіо. Програмне забезпечення Foobar призначене для тих, хто прагне мати повний контроль над прослуховуванням аудіофайлів, не потребуючи яскравого дизайну чи зайвих ефектів. Програмне забезпечення Foobar залишається одним із найбільш гнучких та потужних рішень у сфері відтворення музики, орієнтованим насамперед на якість звуку та функціональність [8]-[11], [14].

Програмне забезпечення Foobar має такі особливості [8]-[11], [14]:

– багатоформатне відтворення: програмне забезпечення Foobar підтримує велику кількість аудіоформатів, включаючи як популярні, так і рідкісні типи файлів [8]-[11], [14];

- налаштовуваний інтерфейс: програмне забезпечення Foobar дозволяє користувачам повністю змінювати вигляд програвача під його особисті потреби [8]-[11], [14];
- підтримка плагінів: програмне забезпечення Foobar підтримує розширення функціональності через систему додаткових компонентів, які можуть бути завантажені окремо [8]-[11], [14];
- зручне тегування: програмне забезпечення Foobar підтримує точне редагування метаданих аудіофайлів і організацію бібліотеки [8]-[11], [14];
- копіювання компакт-дисків: програмне забезпечення Foobar підтримує витягування та збереження аудіо з фізичних носіїв користувача без втрати якості [8]-[11], [14];
- конвертація форматів: програмне забезпечення Foobar дає змогу користувачам перетворювати аудіофайли з одного формату в інший без додаткових програм [8]-[11], [14];
- управління гарячими клавішами: користувач програмного забезпечення Foobar може налаштовувати комбінації клавіш для швидкого керування відтворенням [8]-[11], [14];
- підтримка ReplayGain: програмне забезпечення Foobar забезпечує однаковий рівень гучності під час відтворення різних треків [8]-[11], [14];
- безперервне відтворення: усуває паузи між композиціями для плавного прослуховування аудіофайлів [8]-[11], [14].

Серед недоліків програмного забезпечення Foobar можна відзначити обмежену привабливість інтерфейсу, який виглядає застарілим і може не відповідати очікуванням користувачів, що звикли до сучасного дизайну. Крім того, для реалізації більшості розширених можливостей потрібне встановлення додаткових компонентів, що може бути складним для недосвідчених користувачів. Також слід зазначити мінімальну кількість вбудованих візуалізацій та ефектів, через що відтворення аудіофайлів може здаватися менш інтерактивним у порівнянні з іншими мультимедійними програвачами [8]-[11], [14].

Порівняльну характеристику інформаційних систем для відтворення аудіофайлів наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння інформаційних систем для відтворення аудіофайлів

Критерій порівняння	Dopamine [12]	Clementine [13]	Foobar [14]
Простота інтерфейсу	+	+—	—
Підтримка великої кількості звуковий форматів	+—	+	+
Відтворення аудіо з архівів	—	+—	+
Конвертація аудіо	—	+	+
Візуалізація	+—	+	—
Підтримка потокового аудіо	—	+	+—

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато інформаційних систем для відтворення аудіофайлів. Проте деякі інформаційні системи та програмні засоби для відтворення аудіофайлів мають ряд недоліків, які можуть впливати на їх ефективність та зручність використання. Одним із головних проблем є обмежена підтримка форматів, адже не всі програми здатні працювати з усіма популярними або спеціалізованими форматами аудіофайлів. Це може вимагати додаткових конвертацій або встановлення спеціальних плагінів, що значно ускладнює процес використання. Ще одним важливим недоліком є високі системні вимоги, які можуть призводити до погіршення продуктивності інших застосунків або навіть самого пристрою, особливо на старих чи малопотужних пристроях. Багато програм також мають надто простий інтерфейс, що не завжди підходить користувачам, яким важливі додаткові налаштування та можливість персоналізації програми. Деякі програми також

не надають достатньої якості звуку, особливо на дешевих або старих пристроях, через погану оптимізацію або стиснення звукових файлів, що може негативно вплинути на досвід користувача. Тому актуальною є розробка програмного забезпечення інформаційної системи для відтворення аудіофайлів.

При розробці програмного забезпечення інформаційної системи для відтворення аудіофайлів необхідно забезпечити такі функціональні вимоги:

- підтримка можливості відтворення аудіофайлів;
- підтримка можливості вибору пристрою для відтворення аудіофайлів;
- можливість вибору характеристик звуку для відтворення;
- можливість трансляції звукових потоків іншим користувачам;
- підтримка можливості відтворення звуку, що транслюється іншими користувачами програми;
- підтримка можливості організатора звукової трансляції керування можливістю прослуховування трансляції іншими користувачами.

1.3 Висновки за розділом 1

Визначено, що програмне забезпечення для відтворення аудіофайлів призначено для прослуховування музики чи будь-яких інших звукових записів у цифровому форматі. Таке ПЗ забезпечує підтримку різноманітних аудіоформатів, дозволяє управляти списками відтворення, налаштовувати якість звучання, а також може мати функції потокового передавання або синхронізації з іншими сервісами.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато інформаційних систем для відтворення аудіофайлів. Проте деякі програмні засоби для відтворення аудіофайлів мають ряд недоліків, які можуть впливати на їх ефективність та зручність використання. Одним із головних проблем є обмежена підтримка форматів, адже не всі програми здатні працювати з усіма популярними або спеціалізованими форматами

аудіофайлів. Це може вимагати додаткових конвертацій або встановлення спеціальних плагінів, що значно ускладнює процес використання. Ще одним важливим недоліком є високі системні вимоги, які можуть призводити до погіршення продуктивності інших застосунків або навіть самого пристрою, особливо на старих чи малопотужних пристроях. Багато програм також мають надто простий інтерфейс, що не завжди підходить користувачам, яким важливі додаткові налаштування та можливість персоналізації програми. Деякі програми також не надають достатньої якості звуку, особливо на дешевих або старих пристроях, через погану оптимізацію або стиснення звукових файлів, що може негативно вплинути на досвід користувача. Тому актуальною є розробка програмного забезпечення інформаційної системи для відтворення аудіофайлів.

Сформульовано функціональні вимоги до програмного забезпечення інформаційної системи для відтворення аудіофайлів.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмного забезпечення інформаційної системи для відтворення аудіофайлів було використано мову програмування C# [15], [16].

Програмування на мові C# стало особливо привабливим завдяки гнучкості у створенні різнопланового програмного забезпечення та зручності синтаксису, що сприяє пришвидшенню розробки. Ця мова з'явилась у рамках ініціативи Microsoft, орієнтованої на платформу .NET, і від початку свого існування демонструє високу сумісність із Windows-середовищем, що забезпечило її популярність серед розробників прикладних програм [15], [16].

Сьогодні C# активно застосовується не лише в проєктах для персональних комп'ютерів, а й у сфері мобільного програмування, створенні вебзастосунків та хмарних сервісів. Завдяки впровадженню міжплатформових технологій, зокрема .NET Core та наступних версій загального середовища виконання, стало можливим використовувати єдину кодову базу на різних операційних системах, що значно підвищило ефективність створення універсальних рішень [15], [16].

У синтаксисі мови програмування C# передбачено підтримку актуальних парадигм розробки: можна реалізовувати асинхронність, використовувати лаконічні лямбда-вирази, узагальнення, мовні конструкції для роботи з даними через LINQ. Інструментарій мови програмування C# дозволяє вести глибоке налагодження, що позитивно впливає на надійність програмного продукту [15], [16].

Завдяки постійним оновленням з боку Microsoft, мова програмування C# отримує сучасні мовні можливості, адаптується до нових стандартів і забезпечує тісну інтеграцію з інфраструктурою Azure. Водночас у розвитку мови програмування C# зберігається спрямованість на підвищення доступності мови як для професійних задач, так і для індивідуального використання. Такий підхід робить мову C# доцільною основою для створення

надійних систем, зокрема і в контексті застосунків, пов'язаних із обробкою аудіо або іншими спеціалізованими функціями [15], [16].

При створенні програмного забезпечення для інформаційної системи, пов'язаної з обробкою аудіофайлів, доцільно орієнтуватися на застосування мови програмування, яка забезпечує баланс між потужністю, зручністю та широкими функціональними можливостями. Одним із таких інструментів виступає C#, яка завдяки своїй структурі дає змогу проєктувати складні системи з чіткою логікою та добре організованим кодом. Високий рівень абстракції цієї мови дає змогу ефективно взаємодіяти з компонентами, що відповідають за керування медіа, подіями та списками відтворення [15], [16].

Завдяки вбудованій інтеграції з .NET-платформами, включно з найновішими редакціями середовища виконання, відкривається доступ до надійного набору бібліотек, які можуть бути адаптовані до задач, пов'язаних із аудіо обробкою. Можливість використання спеціалізованих засобів, таких як NAudio або CSCore, спрощує реалізацію аудіофункцій без необхідності створювати все з нуля [15], [16].

Також варто враховувати, що побудова графічного інтерфейсу за допомогою мови програмування C# реалізується без надмірної складності завдяки наявним засобам, які підтримують створення як класичних настільних застосунків, так і рішень для кількох платформ. Робота з багатопотоковістю дозволяє розділяти навантаження між процесами, забезпечуючи одночасну роботу інтерфейсу та обробки аудіоданих, що позитивно впливає на загальну продуктивність системи [15], [16].

Оскільки мова програмування C# активно підтримується спільнотою та постачальником, для розробника доступна велика кількість матеріалів, які можуть допомогти як на початкових етапах, так і при реалізації складніших функцій. Це зменшує час на пошук рішень та полегшує розгортання готового продукту. З огляду на все це, використання C# у якості основної мови для побудови програмного забезпечення системи відтворення аудіо файлів виглядає цілком обґрунтованим і технічно виправданим вибором [15], [16].

Обґрунтування вибору мови програмування для розробки програмного забезпечення інформаційної системи для відтворення аудіофайлів наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки програмного забезпечення інформаційної системи для відтворення аудіофайлів

Критерій порівняння мов програмування	Мова програмування		
	C#	Python	Java
Наявність мультимедійних бібліотек	+	+	+—
Оптимізація роботи з потоками та фоновими задачами	+	+—	+
Підтримка асинхронного програмування	+	+	+—
Зручність для створення ПЗ для відтворення аудіофайлів	+	—	—
Простота реалізації аудіофункціоналу	+	+—	—

Отже, для реалізації програмного забезпечення інформаційної системи для відтворення аудіофайлів обрано мову програмування C#, яка характеризується зрозумілим синтаксисом та широкими можливостями для створення програмного забезпечення різного призначення. Крім того, завдяки активній підтримці з боку Microsoft, мова програмування C# постійно вдосконалюється, отримуючи нові синтаксичні конструкції, можливості компіляції та інтеграції з хмарними сервісами.

2.2 Вибір середовища розробки для створення програмного забезпечення інформаційної системи для відтворення аудіофайлів

В якості середовища розробки для створення програмного забезпечення інформаційної системи для відтворення аудіофайлів було обрано середовище MonoDevelop [17], [18].

Під час створення програмного забезпечення, орієнтованого на платформу .NET, доцільним є використання середовища розробки, що забезпечує гнучкість, підтримку кросплатформних рішень та адаптоване до роботи з мовою програмування C#. Одним із таких інструментів виступає MonoDevelop, яке дає змогу реалізовувати програмні проєкти на різних операційних системах без потреби в радикальній зміні коду чи структури застосунку. Завдяки відповідним архітектурним особливостям забезпечується коректна робота на Windows, Linux та macOS, що дозволяє орієнтуватися на різні середовища розгортання [17], [18].

У процесі розробки з використанням середовища MonoDevelop використовуються сучасні підходи до програмування, зокрема об'єктна модель, взаємодія з зовнішніми бібліотеками, підключення до API та системна інтеграція. Це середовище забезпечує належні умови для поєднання різних частин складного програмного комплексу в єдине цілісне рішення. Робота з кодом виконується в інтерфейсі, що адаптований як для досвідчених користувачів, так і для тих, хто лише починає знайомство з інструментами розробки [17], [18].

До складу середовища MonoDevelop входить інтегрований компілятор і засоби налагодження, що дозволяє контролювати якість програмного продукту в режимі реального часу, своєчасно виявляючи неточності чи помилки. Автоматизовано процес підключення зовнішніх модулів та компонентів, що відповідають специфіці проєкту, забезпечуючи відповідність функціоналу заданим технічним вимогам [17], [18].

Оскільки середовище MonoDevelop підтримує взаємодію з Xamarin, з'являється можливість працювати над застосунками для мобільних платформ, використовуючи єдиний кодовий простір. Це оптимізує розробку і сприяє скороченню часу, необхідного для створення та тестування продукту. Таким чином, MonoDevelop розглядається як універсальне середовище, яке поєднує потужність технологій, функціональність і адаптивність до мультимедійних проєктів різної складності [17], [18].

У процесі створення програмного забезпечення, яке обробляє аудіо дані в інформаційних системах, доцільним є використання середовища MonoDevelop, що забезпечує стабільну інтеграцію з мовою C# і платформою .NET. Середовище розробки MonoDevelop, розроблене з урахуванням підтримки мультимедійних можливостей, дозволяє ефективно реалізовувати застосунки з аудіо функціоналом. Завдяки гнучкості інструментарію та адаптації до різних операційних систем, забезпечується висока сумісність створених продуктів [17], [18].

Особливу роль у розробці програмних систем для відтворення аудіофайлів відіграє підтримка одночасної розробки для кількох платформ, зокрема для macOS, Windows і Linux, що значно розширює можливості поширення програмного забезпечення та його використання на різних пристроях. Це дозволяє забезпечити стабільне відтворення аудіо та обробку звукових даних у незалежності від середовища запуску [17], [18].

Інтерфейс розробки середовища MonoDevelop побудований так, щоб оптимізувати взаємодію з кодом, спростити тестування та забезпечити підтримку інтегрованих засобів налагодження. Такий підхід зменшує складність у роботі з технічними деталями та дозволяє більше уваги приділити розробці логіки застосунку та взаємодії з аудіо даними [17], [18].

Інтеграція із зовнішніми аудіо бібліотеками, які спеціалізуються на роботі зі звуковими потоками, забезпечує додаткові можливості при використанні середовища MonoDevelop для реалізації функцій відтворення,

обробки та контролю параметрів звуку. Завдяки цьому зникає потреба в розробці складного низькорівневого функціоналу з нуля [17], [18].

У підсумку, середовище розробки MonoDevelop можна розглядати як ефективний інструмент для створення програм з розширеними аудіо можливостями, де поєднуються технологічна досконалість, кросплатформеність і зручність розробки [17], [18].

Обґрунтування вибору середовища розробки для створення програмного забезпечення інформаційної системи для відтворення аудіофайлів наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення програмного забезпечення інформаційної системи для відтворення аудіофайлів

Критерій порівняння середовищ розробки	Середовища розробки		
	MonoDevelop	Visual Studio	Rider
Підтримка роботи з мультимедіа	+	+—	+—
Простота інтерфейсу	+	+	+—
Кросплатформність	+	—	+
Швидкість запуску та роботи	+	—	+—
Ліцензія та безкоштовність	+	+—	—

Таким чином, для створення програмного забезпечення інформаційної системи для відтворення аудіофайлів обрано середовище розробки MonoDevelop, оскільки воно забезпечує зручні умови для розробки застосунків на мові C#, яка є оптимальним вибором для реалізації функціоналу відтворення аудіо. Крім того, MonoDevelop має інтуїтивно

зрозумілий інтерфейс, підтримку підсвічування синтаксису, автодоповнення коду, інтеграцію з системами контролю версій та налагоджувач, що дозволяє ефективно знаходити та виправляти помилки.

2.3 Висновки за розділом 2

Для реалізації програмного забезпечення інформаційної системи для відтворення аудіофайлів обрано мову програмування C#, яка характеризується зрозумілим синтаксисом та широкими можливостями для створення програмного забезпечення різного призначення. Крім того, завдяки активній підтримці з боку Microsoft, мова програмування C# постійно вдосконалюється, отримуючи нові синтаксичні конструкції, можливості компіляції та інтеграції з хмарними сервісами.

Для створення програмного забезпечення інформаційної системи для відтворення аудіофайлів обрано середовище розробки MonoDevelop, оскільки воно забезпечує зручні умови для розробки застосунків на мові C#, яка є оптимальним вибором для реалізації функціоналу відтворення аудіо. Крім того, MonoDevelop має інтуїтивно зрозумілий інтерфейс, підтримку підсвічування синтаксису, автодоповнення коду, інтеграцію з системами контролю версій та налагоджувач, що дозволяє ефективно знаходити та виправляти помилки.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення інформаційної системи для відтворення аудіофайлів

Структуру програмного забезпечення інформаційної системи для відтворення аудіофайлів наведено на рис. 3.1.

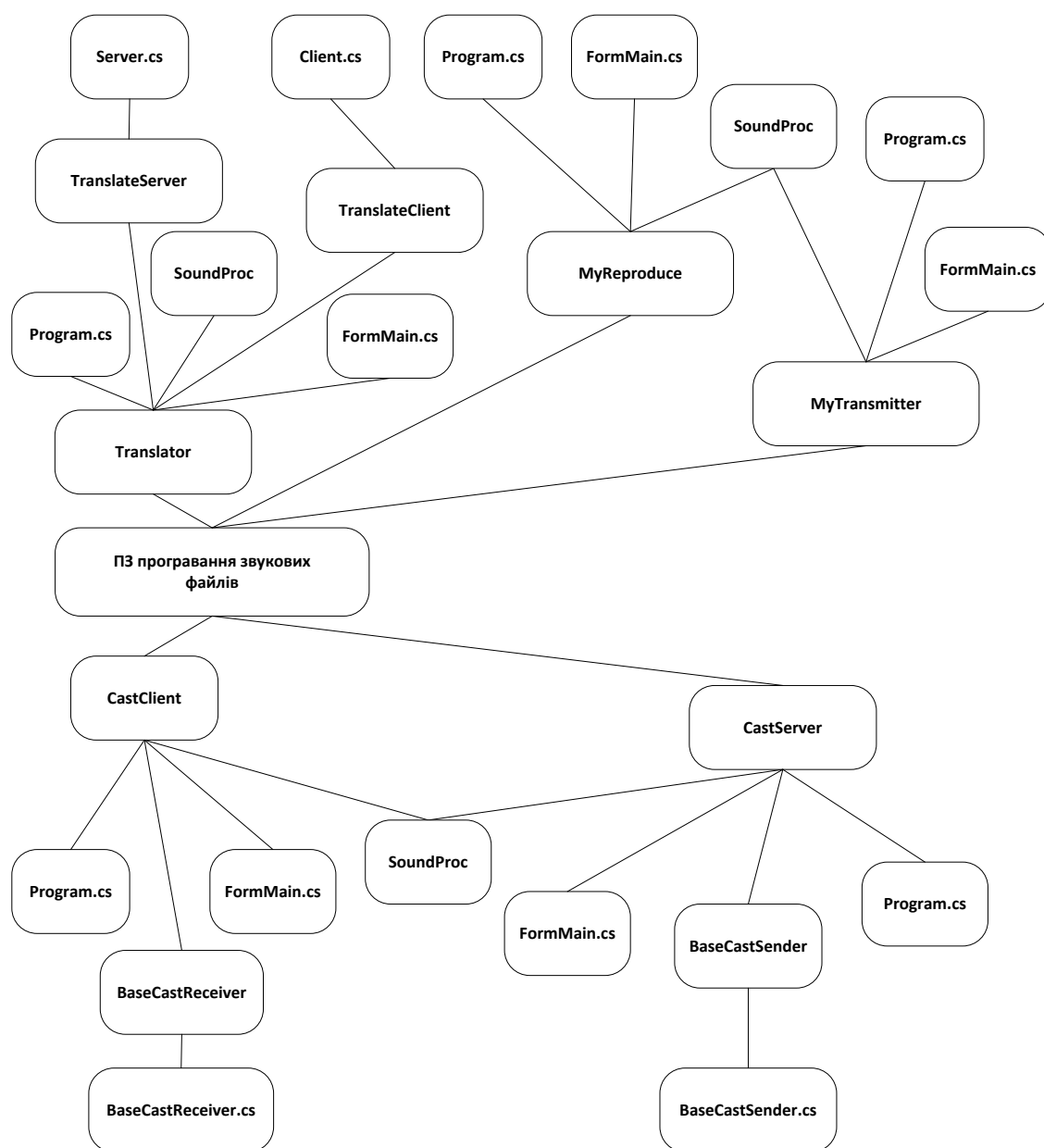


Рисунок 3.1 – Структура програмного забезпечення інформаційної системи для відтворення аудіофайлів

У програмному забезпеченні для відтворення аудіофайлів реалізовано кілька функціональних блоків, які забезпечують як локальне відтворення, так і передачу звукових потоків через мережу. Основу системи становить технологічне ядро SoundProc, що функціонує у вигляді динамічної бібліотеки. Цей модуль відповідає за всі ключові операції зі звуковим потоком: керування буферизацією, обробку протоколів передачі, взаємодію з таймерами та звуковими пристроями, а також за здійснення внутрішніх перетворень форматів аудіо. Усе це дозволяє гнучко адаптувати аудіосигнали до поточних умов відтворення або передачі.

До структури входить компонент MyReproduce, який реалізує механізми прослуховування аудіо. Він орієнтований на кінцевого користувача й використовує функціонал ядра SoundProc. У межах цього модуля основним є програмний файл запуску та візуальна форма, що дає змогу керувати програванням у графічному інтерфейсі.

Інший сегмент системи відтворення аудіофайлів MyTransmitter виконує задачі переспрямування аудіо між джерелами. Як і MyReproduce, він ґрунтується на SoundProc і має аналогічну структуру запуску та інтерфейсу, але призначений для передачі потоків.

Для підтримки передавання аудіо через мережу передбачено компонент Translator. У ньому реалізовано як серверну, так і клієнтську частину трансляції: перша дозволяє здійснювати прийом вхідного трафіку, а друга — його надсилання. Компонент працює з бібліотекою SoundProc та містить інтерфейсну частину, а також модулі Server.cs і Client.cs для відповідної мережевої логіки.

Окремо розроблено підсистему CastServer, яка відповідає за організацію широкомовного передавання аудіо. У межах цього модуля передбачено використання бібліотеки SoundProc, інтерфейс взаємодії з користувачем та спеціалізований модуль BaseCastDispatcher для реалізації багатоточкової передачі.

Для приймання таких широкомовних потоків використовується клієнтський компонент CastClient. Він, подібно до інших, побудований на основі SoundProc, має окремий інтерфейс і використовує модуль BaseCastReceiver для отримання аудіо з широкомовної розсилки.

Кожен із зазначених блоків містить власний вхідний файл Program.cs, що відповідає за ініціалізацію відповідної частини застосунку, і окремі елементи логіки обробки — залежно від призначення (відтворення, передача, трансляція чи прийом). Завдяки такій модульній організації забезпечується розширюваність, підтримка різних сценаріїв використання та масштабування системи для роботи з аудіо в інформаційних системах.

3.2 Функціонування програмного забезпечення інформаційної системи для відтворення аудіофайлів

Схему функціонування програмного забезпечення інформаційної системи для відтворення аудіофайлів наведено на рис. 3.2.

Програмне забезпечення, призначене для роботи з аудіофайлами у межах інформаційної системи, організоване з урахуванням архітектурної моделі клієнт-сервер. Такий підхід обрано через необхідність функціонування в умовах локальної мережі підприємства, де частина обчислень і обробки даних виконується на клієнтських пристроях, а решта – на серверній стороні.

Окремі компоненти програмного забезпечення інформаційної системи для відтворення аудіофайлів, зокрема ті, що відповідають за відтворення звуку (MyReproduce) та його ретрансляцію (MyTransmitter), працюють автономно на робочих станціях користувачів. Їх запуск виконується безпосередньо на комп'ютері, де вони отримують доступ до звукових пристроїв або файлів, і забезпечують необхідні дії користувача без залучення мережесервісів.

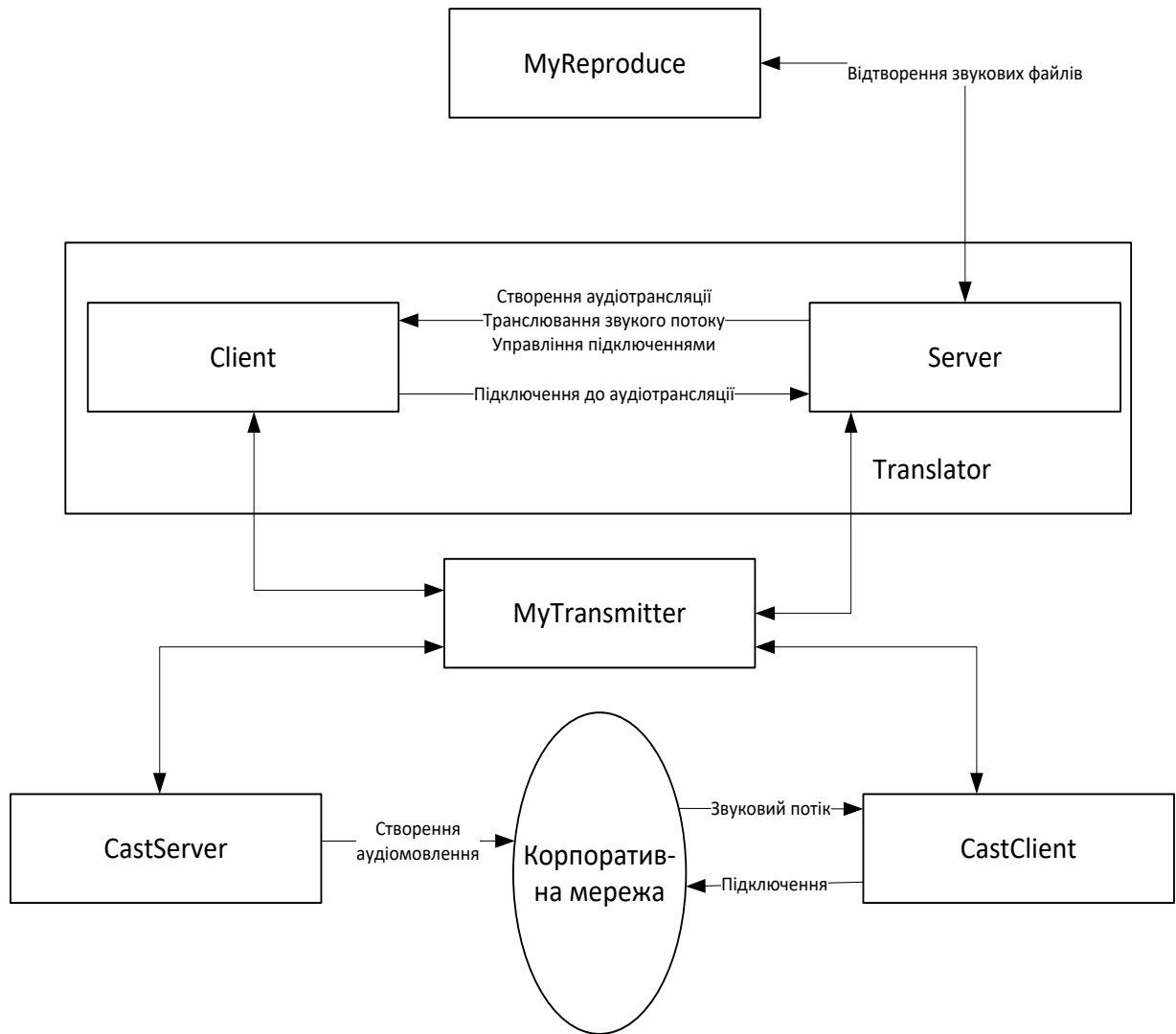


Рисунок 3.2 – Схема функціонування програмного забезпечення інформаційної системи для відтворення аудіофайлів

Решта функціоналу програмного забезпечення інформаційної системи для відтворення аудіофайлів ґрунтується на мережевій взаємодії між клієнтом і сервером. Для забезпечення трансляційного режиму використовується компонент Translator, який здатен виконувати як функції передавача, так і отримувача потокового звуку. Серверна частина цього застосунку ініціює створення нових сеансів передавання аудіо, керує підключенням клієнтів, а також вимагає окремого запуску модулю відтворення (MyReproduce) для обробки звукових файлів. Передавання в цьому режимі організоване за допомогою протоколу TCP.

Функціонал, що відповідає за внутрішнє радіомовлення, представлений двома незалежними застосунками: CastServer і CastClient. Перший здійснює трансляцію, використовуючи як джерело аудіосигнал із локального пристрою або вибраного файлу, другий – приймає цей сигнал і забезпечує його відтворення на стороні користувача. Обмін даними при цьому здійснюється відповідно до специфікацій протоколу RTP, що забезпечує передачу аудіо в режимі реального часу.

Загалом, така схема дозволяє комбінувати локальні та мережеві можливості програмного забезпечення інформаційної системи для відтворення аудіофайлів для реалізації задач як простого програвання звукових даних, так і масштабованого аудіомовлення у корпоративному середовищі.

Функціональність інформаційного програмного забезпечення для відтворення аудіофайлів передбачає реалізацію комплексу дій, що охоплюють відтворення звукового контенту, а також обробку й передавання аудіопотоків у межах локальних мереж і мережі Інтернет. Програмне забезпечення інформаційної системи для відтворення аудіофайлів забезпечує можливість запуску аудіофайлів для прослуховування, ініціації трансляцій у режимі реального часу та участі в них, що дозволяє формувати потокове мовлення. Крім того, підтримується створення нових сесій трансляції з можливістю їх адміністрування, а також підключення до вже активних потоків за умови наявності доступу.

Завдяки таким можливостям програмного забезпечення інформаційної системи для відтворення аудіофайлів здатне виконувати завдання внутрішньої телефонії та мовлення в межах корпоративної інфраструктури. Аудіопотоки можуть використовуватися для оперативного інформування, координації дій співробітників або створення постійного фоново-інформаційного супроводу (аналогічно до радіомовлення). Таким чином, система забезпечує не лише індивідуальне прослуховування, але й повноцінну інтеграцію голосових сервісів у єдину корпоративну мережу.

3.3 Розробка діаграми прецедентів програмного забезпечення інформаційної системи для відтворення аудіофайлів

Діаграма прецедентів для програмного забезпечення, яке забезпечує обробку та передавання аудіофайлів, зображена на рис. 3.3 та демонструє взаємодію користувача із системою через низку доступних сценаріїв використання. Основна увага приділяється можливостям, пов'язаним із керуванням аудіо та трансляціями.

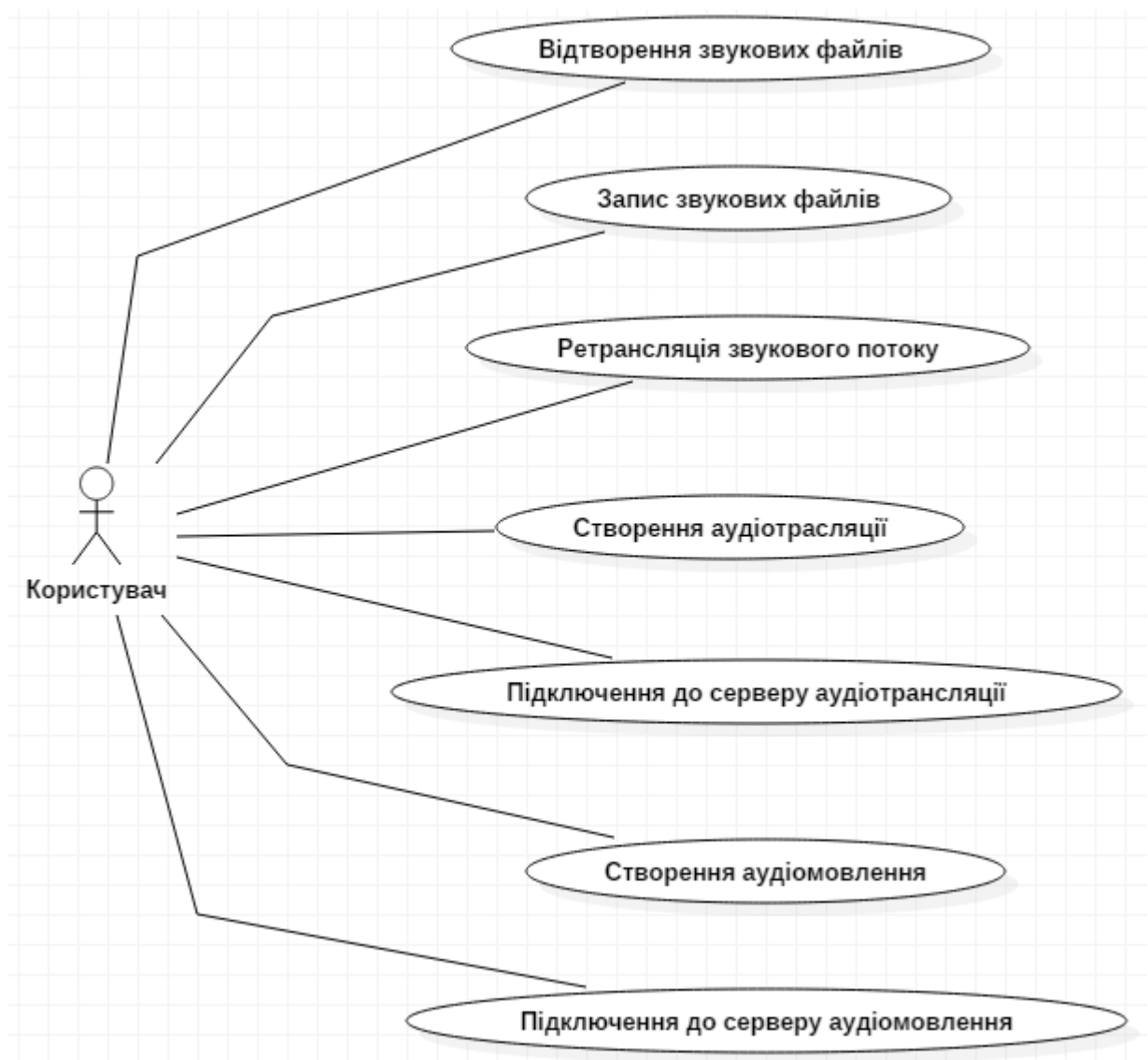


Рисунок 3.3 – Діаграма прецедентів програмного забезпечення інформаційної системи для відтворення аудіофайлів

Користувач взаємодіє з системою для ініціації процесів відтворення локальних звукових файлів. Передбачена також можливість створення нових аудіопотоків шляхом захоплення аудіо й його передавання до інших пристроїв або користувачів. Реалізовано механізм налаштування та запуску трансляцій, що дозволяє системі функціонувати як джерело аудіомовлення.

Окрім цього, передбачено функціонал для управління вже активними трансляційними сесіями: доступне їх завершення, повторне відкриття або внесення змін у параметри потоків. Користувач також має змогу приєднатися до наявних аудіопотоків для їх прослуховування, що дозволяє інтегрувати систему в телефонну або радіотрансляційну мережу всередині організації.

Таким чином, на діаграмі прецедентів візуалізовано основні функції, які доступні кінцевому користувачеві: від простого прослуховування аудіофайлів до керування багатокористувацькими трансляціями, що охоплюють задачі корпоративної телефонії та внутрішнього радіомовлення.

3.4 Проєктування інтерфейсу програмного забезпечення інформаційної системи для відтворення аудіофайлів

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

- підтримка можливості відтворення аудіофайлів;
- підтримка можливості вибору пристрою для відтворення аудіофайлів;
- можливість вибору характеристик звуку для відтворення;
- можливість трансляції звукових потоків іншим користувачам;
- підтримка можливості відтворення звуку, що транслюється іншими користувачами програми;

– підтримка можливості організатора звукової трансляції керування можливістю прослуховування трансляції іншими користувачами.

Інтерфейс головної форми програмного забезпечення інформаційної системи для відтворення аудіофайлів наведено на рис. 3.4.

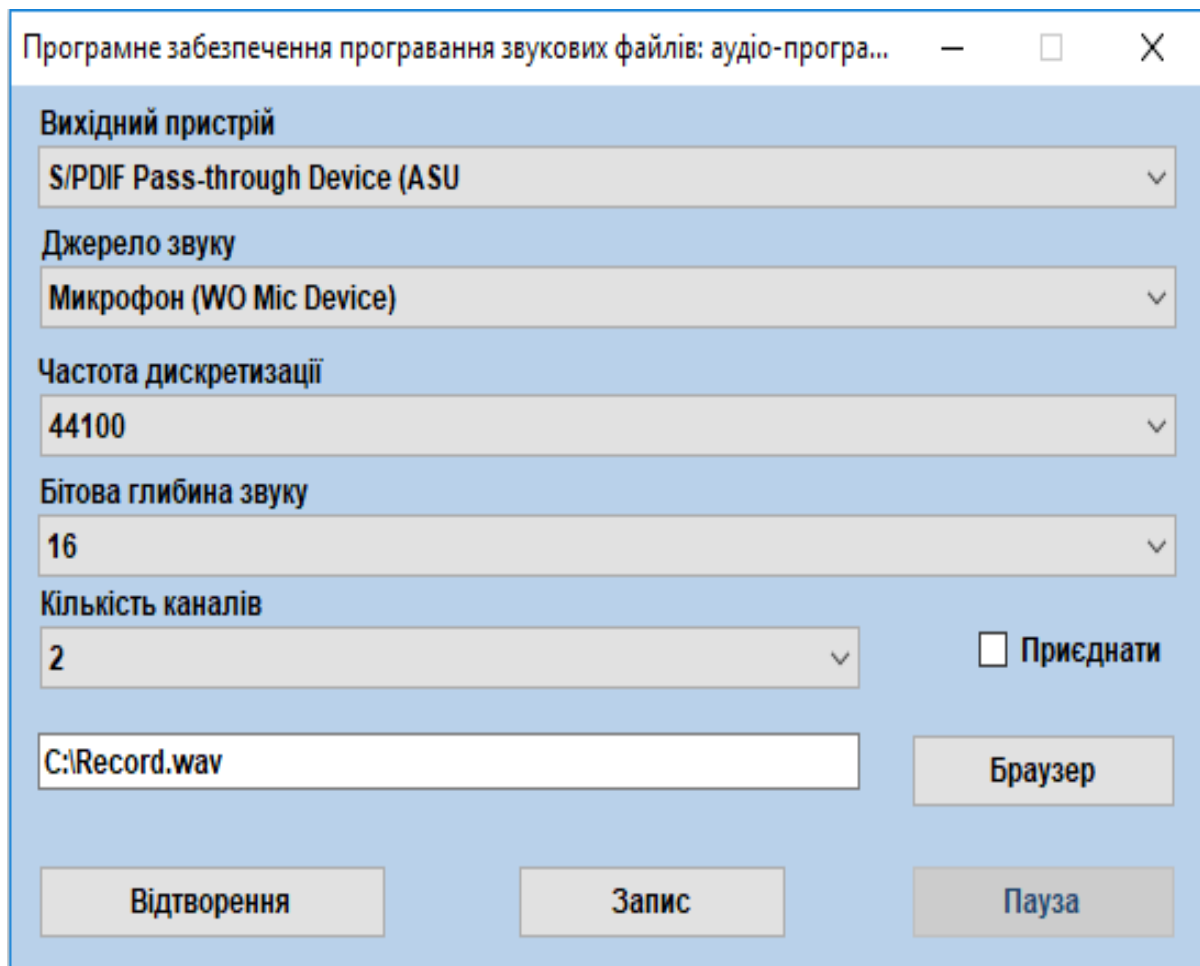


Рисунок 3.4 – Інтерфейс головної форми програмного забезпечення інформаційної системи для відтворення аудіофайлів

Форма програмного забезпечення для відтворення аудіофайлів у режимі трансляції аудіо потоку наведена на рис. 3.5.

Форма програмного забезпечення для відтворення аудіофайлів у режимі трансляції аудіо потоку на стороні клієнта наведена на рис. 3.6.

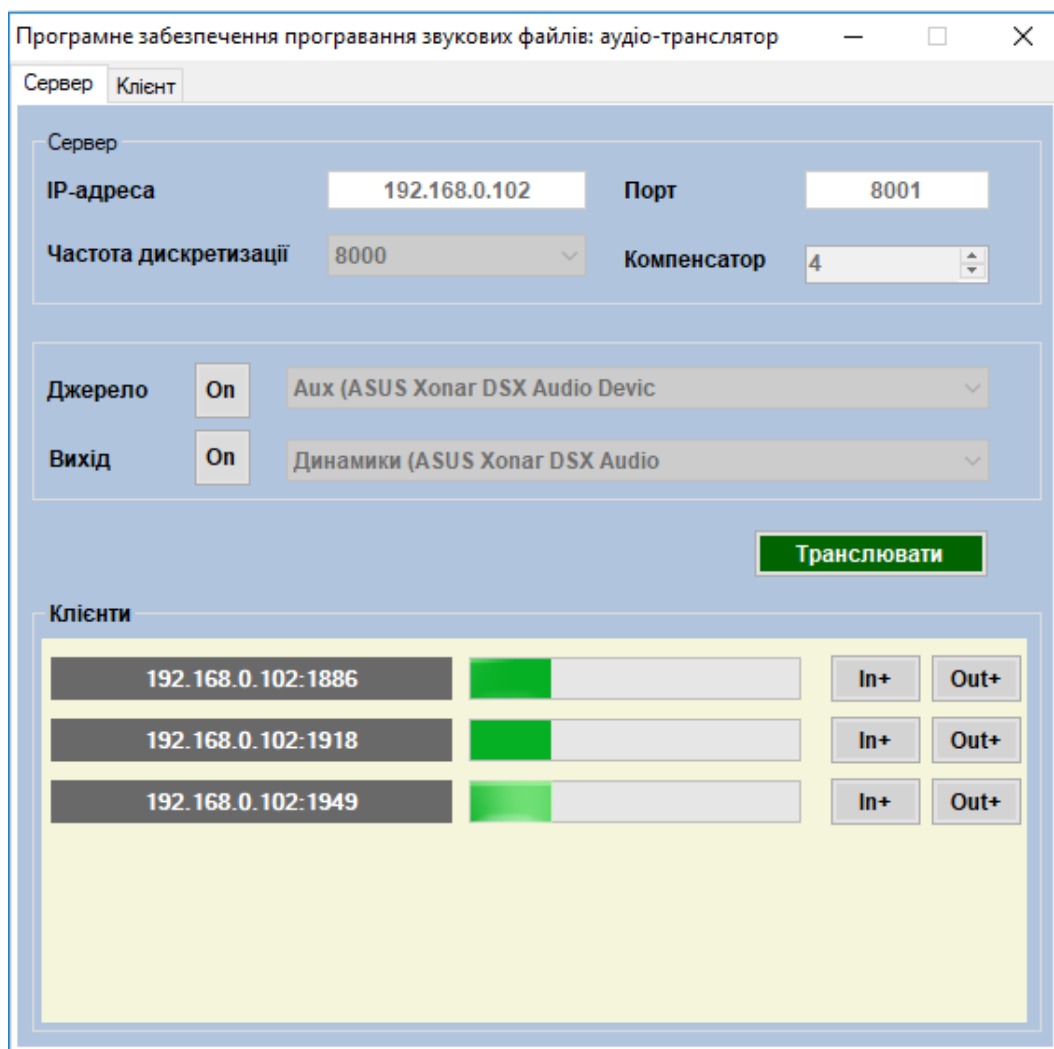


Рисунок 3.5 – Вікно роботи з аудіо трансляцією на сервері

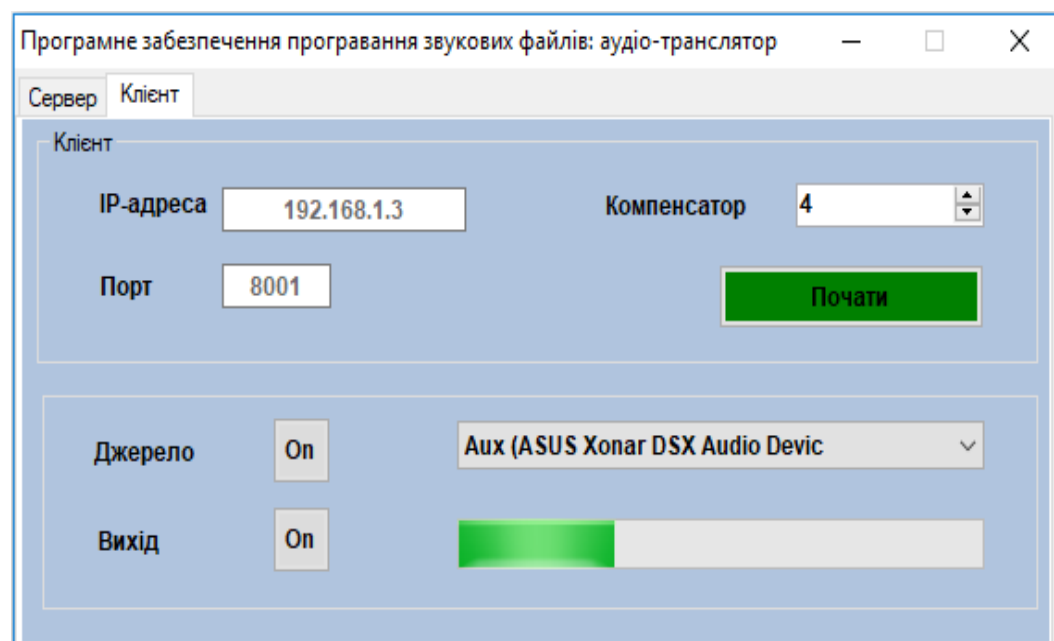


Рисунок 3.6 – Вікно роботи з аудіо трансляцією на стороні клієнта

3.5 Висновки за розділом 3

Розроблено програмне забезпечення інформаційної системи для відтворення аудіофайлів.

Запропоновано структуру програмного забезпечення інформаційної системи для відтворення аудіофайлів. Визначено, що у програмному забезпеченні для відтворення аудіофайлів реалізовано кілька функціональних блоків, які забезпечують як локальне відтворення, так і передачу звукових потоків через мережу. Завдяки такій модульній організації забезпечується розширюваність, підтримка різних сценаріїв використання та масштабування системи для роботи з аудіо в інформаційних системах.

Розроблено схему функціонування програмного забезпечення інформаційної системи для відтворення аудіофайлів, що дозволяє комбінувати локальні та мережеві можливості програмного забезпечення для реалізації задач як простого програвання звукових даних, так і масштабованого аудіомовлення у корпоративному середовищі.

Запропоновано діаграму прецедентів програмного забезпечення для відтворення аудіофайлів, що демонструє взаємодію користувача із системою через низку доступних сценаріїв використання, від простого прослуховування аудіофайлів до керування багатокористувацькими трансляціями, що охоплюють задачі корпоративної телефонії та внутрішнього радіомовлення.

Описано особливості реалізації програмного забезпечення інформаційної системи для відтворення аудіофайлів. Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням інформаційної системи для відтворення аудіофайлів.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для підтримки процесу відтворення аудіофайлів. Програма написана на мові C#, яка характеризується зрозумілим синтаксисом та широкими можливостями для створення програмного забезпечення різного призначення. В якості середовища розробки для створення програмного забезпечення для відтворення аудіофайлів обрано MonoDevelop, оскільки воно забезпечує зручні умови для розробки застосунків на мові C#, яка є оптимальним вибором для реалізації функціоналу відтворення аудіо. Крім того, MonoDevelop має інтуїтивно зрозумілий інтерфейс, підтримку підсвічування синтаксису, автодоповнення коду, інтеграцію з системами контролю версій та налагоджувач, що дозволяє ефективно знаходити та виправляти помилки.

Програмне забезпечення інформаційної системи для відтворення аудіофайлів забезпечує виконання таких функцій:

- підтримка можливості відтворення аудіофайлів;
- підтримка можливості вибору пристрою для відтворення аудіофайлів;
- можливість вибору характеристик звуку для відтворення;
- можливість трансляції звукових потоків іншим користувачам;
- підтримка можливості відтворення звуку, що транслюється іншими користувачами програми;
- підтримка можливості організатора звукової трансляції керування можливістю прослуховування трансляції іншими користувачами.

Для роботи програмного забезпечення інформаційної системи для відтворення аудіофайлів необхідно забезпечити відповідний рівень технічного та програмного оснащення. Серед апаратних вимог – наявність процесора, не із частотою не менше 500 МГц, а також обсяг оперативної пам'яті щонайменше 2 гігабайти, хоча для стабільної роботи системи бажано мати

більше. Жорсткий диск має містити вільний простір у розмірі щонайменше 500 мегабайт для інсталяції основних компонентів, при цьому бажано мати додаткове місце для тимчасового зберігання аудіо-даних. Аудіосистема комп'ютера повинна включати звукову карту. Також потрібен стандартний мережевий адаптер, який забезпечує доступ до локальної мережі для передачі аудіопотоків.

З програмного боку, система повинна функціонувати під керуванням операційної системи, наприклад, Windows або дистрибутива Linux, що підтримує звукові підсистеми типу ALSA чи PulseAudio. Для створення аудіо-трансляцій на серверній стороні необхідно встановити додаток, який підтримує передачу звукових потоків за протоколом TCP, а також мати засіб для ретрансляції звуку до інших пристроїв або комп'ютерів у мережі.

На клієнтських комп'ютерах мають бути встановлені компоненти, що дозволяють приймати TCP-трансляції для реалізації дзвінків або корпоративного аудіомовлення. При потребі користувач також може скористатися стандартним програвачем для прослуховування окремих аудіофайлів.

Функціональні характеристики розробленого програмного забезпечення інформаційної системи для відтворення аудіофайлів наведено у технічному завданні (додаток А). Фрагмент тексту програмного забезпечення інформаційної системи для відтворення аудіофайлів наведено у додатку Б.

4.2 Характеристики програмного забезпечення інформаційної системи для відтворення аудіофайлів

Програмне забезпечення інформаційної системи для відтворення аудіофайлів складається з кількох функціональних блоків, які забезпечують як локальне відтворення, так і передачу звукових потоків через мережу. Завдяки такій модульній організації забезпечується розширюваність, підтримка різних

сценаріїв використання та масштабування системи для роботи з аудіо в інформаційних системах.

Програмне забезпечення для відтворення аудіофайлів характеризується підтримкою основних форматів звукових даних, що дозволяє забезпечити сумісність з різними аудіофайлами. Його робота ґрунтується на здатності приймати, обробляти та виводити аудіосигнал із локального джерела або з мережі. Застосовується механізм потокової передачі даних, що дозволяє забезпечити безперервне звучання навіть у разі використання мережевих ресурсів.

Програмне забезпечення підтримує модульну архітектуру, що дає змогу відокремлено керувати компонентами для відтворення, ретрансляції та прийому аудіо. Використовується клієнт-серверна модель, завдяки якій здійснюється централізоване управління трансляціями в межах локальної мережі. Для передачі звукових потоків застосовуються стандартні мережеві протоколи, що дає змогу інтегрувати систему в існуючу ІТ-інфраструктуру.

Забезпечується гнучкість налаштувань, зокрема можливість обирати джерело звуку, призначення вихідного аудіопотоку, змінювати параметри звукового потоку залежно від технічних умов. Передбачено можливість одночасної роботи декількох екземплярів програм на різних вузлах мережі. Інтерфейс користувача реалізовано таким чином, щоб забезпечити інтуїтивне керування процесами трансляції, відтворення та підключення до існуючих сеансів мовлення.

Повноцінна робота програмної системи передбачає стабільне мережеве з'єднання, можливість одночасного запуску кількох застосунків і, при необхідності, інтеграцію з вебінтерфейсом через сучасний браузер.

Передбачено автоматичне відновлення зв'язку у разі мережевих збоїв. Підтримується сумісність із базовими звуковими підсистемами операційних систем, що забезпечує стабільність і відтворення звуку без додаткових драйверів.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску програмного забезпечення інформаційної системи для відтворення аудіофайлів необхідно запустити виконуваний файл.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення інформаційної системи для відтворення аудіофайлів є звукові файли або звукові потоки.

Вихідними даними програмного забезпечення інформаційної системи для відтворення аудіофайлів є звуковий потік, який відтворюється на пристрої (звуковій карті) користувача.

4.3.3 Повідомлення

Користувач програмного забезпечення інформаційної системи для відтворення аудіофайлів може отримати повідомлення про некоректно обраний файл для прослуховування звуку та інші повідомлення.

4.4 Виконання програмного забезпечення інформаційної системи для відтворення аудіофайлів

Після запуску програмного забезпечення інформаційної системи для відтворення аудіофайлів користувачу відображається головна форма (рис. 4.1). Для вибору звукового файлу з метою його подальшого відтворення (програвання) необхідно або ввести його шлях, або натиснути кнопку Браузер та у діалоговому вікні, що з'явилося, обрати відповідний файл. Для програвання звукового файлу необхідно натиснути кнопку Відтворення.

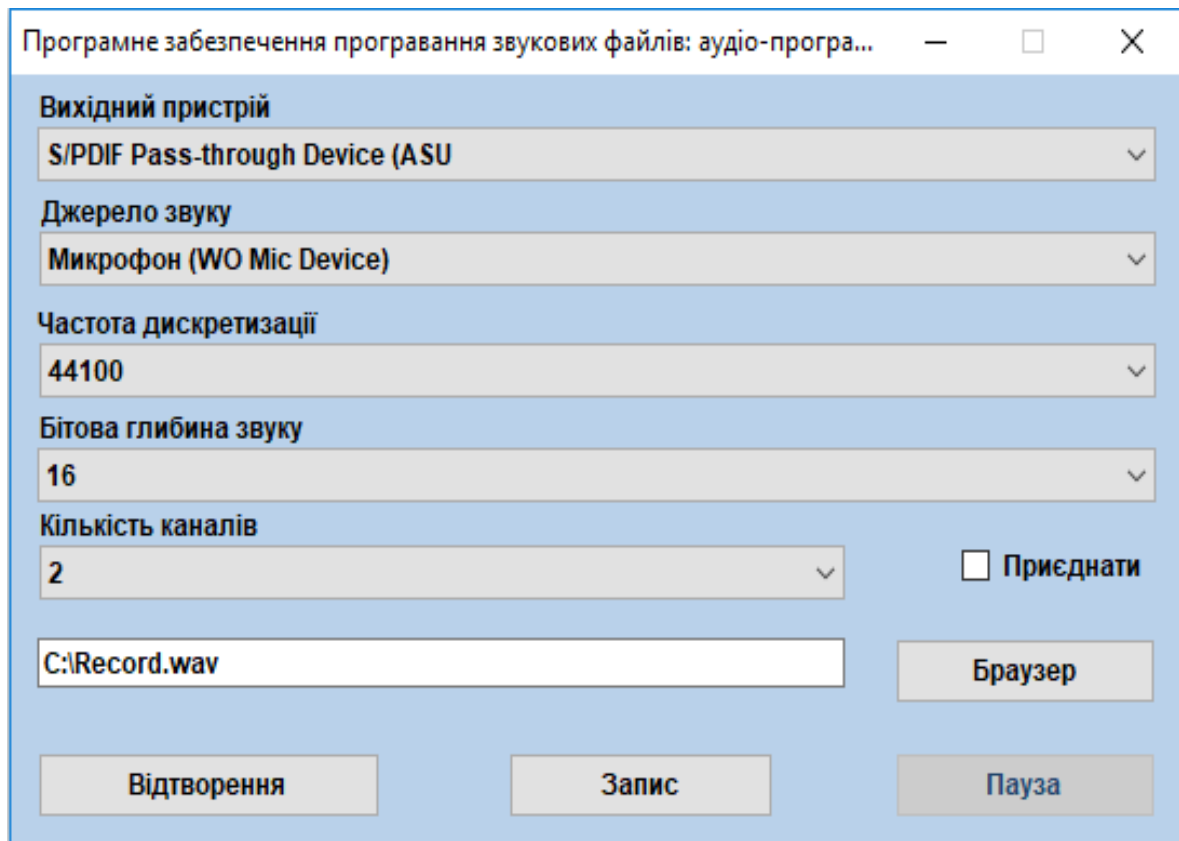


Рисунок 4.1 – Інтерфейс головної форми програмного забезпечення для відтворення аудіофайлів

Також на головній формі міститься інформація про доступні аудіо входи (джерело звуку) та аудіо виходи (вихідний пристрій), частоту дискретизація та інші параметри відтворення звуку.

Форма програмного забезпечення для відтворення аудіофайлів у режимі трансляції аудіо потоку наведена на рис. 4.2. У цьому режимі програма перенаправляє існуючий аудіо потік іншим користувачам. Джерелом звуку може бути аудіофайл або мікрофон, підключений до комп'ютера, що виконує функції сервера (транслятора). Для включення мікрофону необхідно натиснути кнопку **Он**. Для початку трансляції натискається кнопка **Транслювати**. На вкладці **Сервер** задаються також й інші параметри, необхідні для трансляції звуку. Вибір пристрою прослуховування відтвореного звуку трансляції здійснюється за допомогою списку, що відповідає полю **Вихід**.

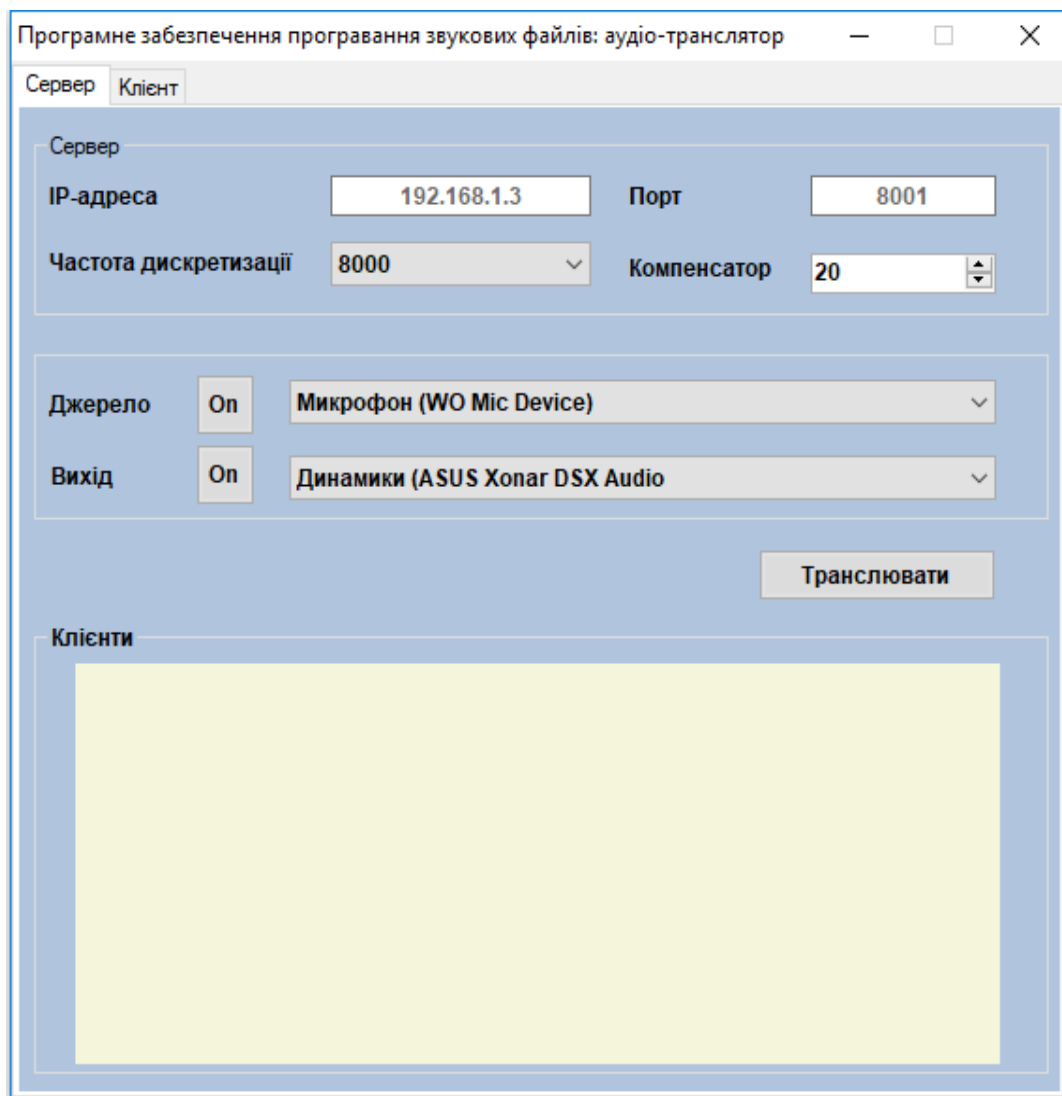


Рисунок 4.2 – Вікно роботи з аудіо трансляцією на сервері

Форма програмного забезпечення для відтворення аудіофайлів у режимі трансляції аудіо потоку на стороні клієнта наведена на рис. 4.3. Тут користувач обирає основні параметри трансляції та натискає кнопку Почати.

Під час трансляції звукових даних в області Клієнти вкладки Сервер (рис. 4.4) відображається список клієнтів (користувачів), які прослуховують трансляцію, що здійснюється сервером. Користувач Сервера може вимикати або вмикати джерела звуку та пристроїв його відтворення на пристроях клієнтів за допомогою кнопок In+ та Out+. Таким чином організатор звукової трансляції має можливість керування прослуховуванням трансляції іншими користувачами.

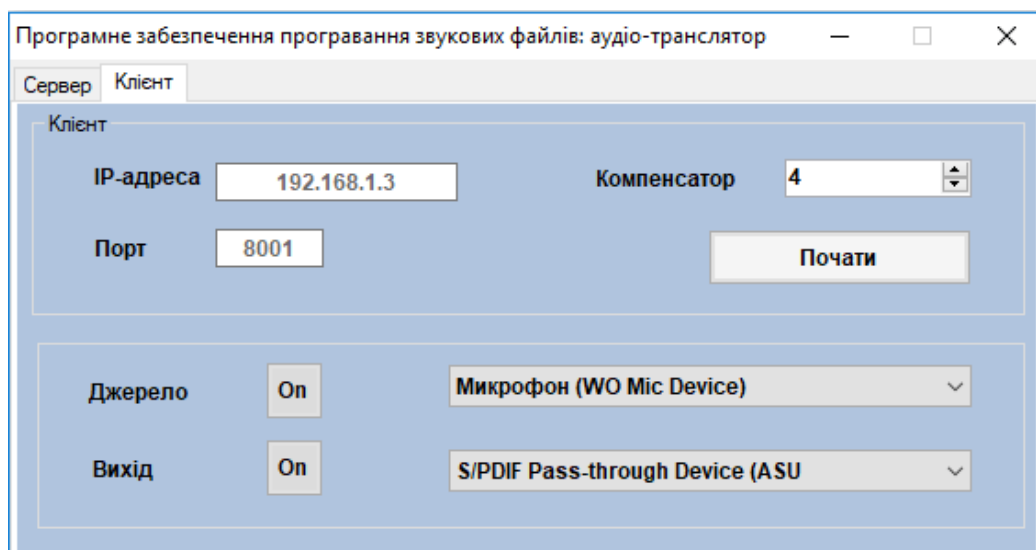


Рисунок 4.3 – Вікно роботи з аудіо трансляцією на стороні клієнта

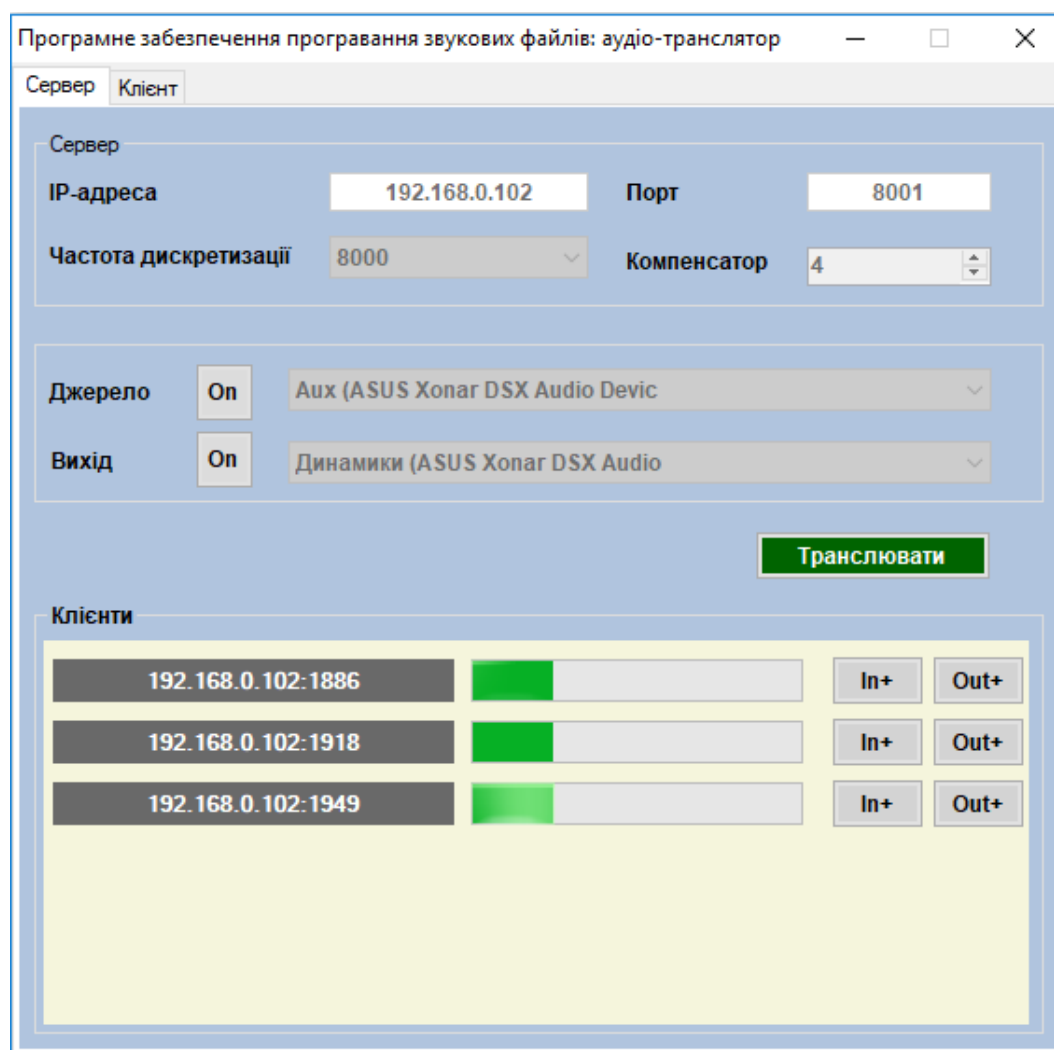


Рисунок 4.4 – Список підключених клієнтів до сервера програмного забезпечення відтворення аудіофайлів

Під час трансляції та відтворення звуку на стороні клієнта відображається інформація про джерело аудіо даних (рис. 4.5).

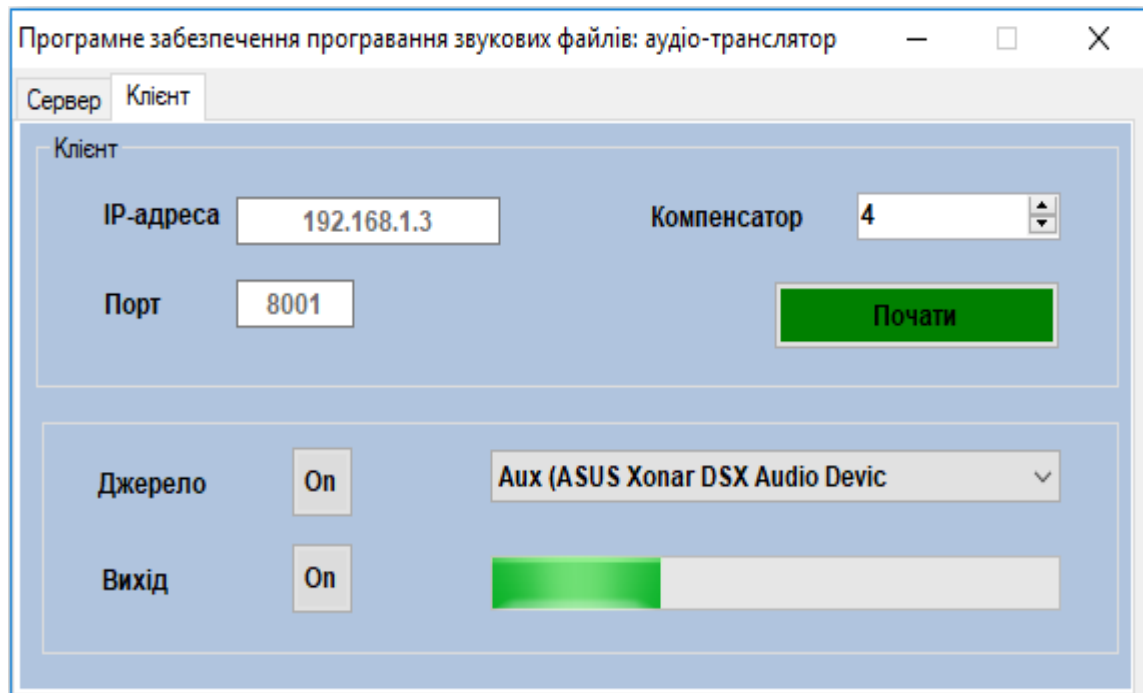


Рисунок 4.5 – Інформація про джерело аудіо даних на стороні клієнта

4.5 Тестування програмного забезпечення інформаційної системи для відтворення аудіофайлів

Проведено тестування програмного забезпечення інформаційної системи для відтворення аудіофайлів.

Підтверджено, що програмне забезпечення інформаційної системи для відтворення аудіофайлів працює коректно і стабільно, виконуючи всі необхідні функціональні вимоги та успішно справляючись із основною задачею підтримки процесу прослуховування музики та інших аудіо файлів.

Розроблене програмне забезпечення забезпечує автоматизацію операцій інформаційної системи, пов'язаних із підтримкою процесів відтворення аудіофайлів.

4.6 Висновки за розділом 4

Описано програмне забезпечення інформаційної системи для відтворення аудіофайлів.

Проведено тестування створеного програмного забезпечення інформаційної системи для відтворення аудіофайлів. Результати тестування показали, що програмне забезпечення інформаційної системи для відтворення аудіофайлів забезпечує автоматизацію процесів, пов'язаних із підтримкою відтворення аудіофайлів.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки засобів для відтворення аудіофайлів.

Визначено, що програмне забезпечення для відтворення аудіофайлів призначено для прослуховування музики чи будь-яких інших звукових записів у цифровому форматі. Таке ПЗ забезпечує підтримку різноманітних аудіоформатів, дозволяє управляти списками відтворення, налаштовувати якість звучання, а також може мати функції потокового передавання або синхронізації з іншими сервісами.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато інформаційних систем для відтворення аудіофайлів. Проте деякі програмні засоби для відтворення аудіофайлів мають ряд недоліків, які можуть впливати на їх ефективність та зручність використання. Одним із головних проблем є обмежена підтримка форматів, адже не всі програми здатні працювати з усіма популярними або спеціалізованими форматами аудіофайлів. Це може вимагати додаткових конвертацій або встановлення спеціальних плагінів, що значно ускладнює процес використання. Ще одним важливим недоліком є високі системні вимоги, які можуть призводити до погіршення продуктивності інших застосунків або навіть самого пристрою, особливо на старих чи малопотужних пристроях. Багато програм також мають надто простий інтерфейс, що не завжди підходить користувачам, яким важливі додаткові налаштування та можливість персоналізації програми. Деякі програми також не надають достатньої якості звуку, особливо на дешевих або старих пристроях, через погану оптимізацію або стиснення звукових файлів, що може негативно вплинути на досвід користувача. Тому актуальною є розробка програмного забезпечення інформаційної системи для відтворення аудіофайлів.

Сформульовано функціональні вимоги до програмного забезпечення інформаційної системи для відтворення аудіофайлів.

Для реалізації програмного забезпечення інформаційної системи для відтворення аудіофайлів обрано мову програмування C#, яка характеризується зрозумілим синтаксисом та широкими можливостями для створення програмного забезпечення різного призначення. Крім того, завдяки активній підтримці з боку Microsoft, мова програмування C# постійно вдосконалюється, отримуючи нові синтаксичні конструкції, можливості компіляції та інтеграції з хмарними сервісами.

Для створення програмного забезпечення інформаційної системи для відтворення аудіофайлів обрано середовище розробки MonoDevelop, оскільки воно забезпечує зручні умови для розробки застосунків на мові C#, яка є оптимальним вибором для реалізації функціоналу відтворення аудіо. Крім того, MonoDevelop має інтуїтивно зрозумілий інтерфейс, підтримку підсвічування синтаксису, автодоповнення коду, інтеграцію з системами контролю версій та налагоджувач, що дозволяє ефективно знаходити та виправляти помилки.

Розроблено програмне забезпечення інформаційної системи для відтворення аудіофайлів.

Запропоновано структуру програмного забезпечення інформаційної системи для відтворення аудіофайлів. Визначено, що у програмному забезпеченні для відтворення аудіофайлів реалізовано кілька функціональних блоків, які забезпечують як локальне відтворення, так і передачу звукових потоків через мережу. Завдяки такій модульній організації забезпечується розширюваність, підтримка різних сценаріїв використання та масштабування системи для роботи з аудіо в інформаційних системах.

Розроблено схему функціонування програмного забезпечення інформаційної системи для відтворення аудіофайлів, що дозволяє комбінувати локальні та мережеві можливості програмного забезпечення для реалізації

задач як простого програвання звукових даних, так і масштабованого аудіомовлення у корпоративному середовищі.

Запропоновано діаграму прецедентів програмного забезпечення для відтворення аудіофайлів, що демонструє взаємодію користувача із системою через низку доступних сценаріїв використання, від простого прослуховування аудіофайлів до керування багатокористувацькими трансляціями, що охоплюють задачі корпоративної телефонії та внутрішнього радіомовлення.

Описано особливості реалізації програмного забезпечення інформаційної системи для відтворення аудіофайлів. Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням інформаційної системи для відтворення аудіофайлів.

Проведено тестування створеного програмного забезпечення інформаційної системи для відтворення аудіофайлів. Результати тестування показали, що програмне забезпечення інформаційної системи для відтворення аудіофайлів забезпечує автоматизацію процесів, пов'язаних із підтримкою відтворення аудіофайлів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. How to Make a Music App: A Comprehensive Guide [Electronic resource]. – Access mode: <https://anyforsoft.com/blog/create-a-music-app/>.
2. Lessons Learned from Creating a Simple Audio Player [Electronic resource]. – Access mode: <https://jatinchowdhury18.medium.com/lessons-learned-from-creating-a-simple-audio-player-656ade200216>.
3. Tutorial: Build an audio player [Electronic resource]. – Access mode: https://juce.com/tutorials/tutorial_playing_sound_files/.
4. A Step by Step Guide to Building an Android Audio Player App [Electronic resource]. – Access mode: <https://www.sitepoint.com/a-step-by-step-guide-to-building-an-android-audio-player-app/>.
5. How to create a custom audio player for your FlutterFlow app [Electronic resource]. – Access mode: <https://www.rapidevelopers.com/flutterflow-tutorials/how-to-create-a-custom-audio-player-for-your-flutterflow-app>.
6. What are some common audio players [Electronic resource]. – Access mode: https://fileinfo.com/help/audio_players.
7. Building a Music Player GUI Application with Python [Electronic resource]. – Access mode: <https://devnithw.medium.com/building-a-music-player-gui-application-with-python-7c3b03f812fc>.
8. Best Music Players for Windows [Electronic resource]. – Access mode: <https://helpdeskgeek.com/9-best-music-players-for-windows-11/>.
9. Best Free Audio Players for Windows [Electronic resource]. – Access mode: <https://www.thepcinsider.com/best-free-audio-players-windows/>.
10. What is the best digital music player software [Electronic resource]. – Access mode: <https://www.slant.co/topics/6134/~digital-music-player-software>.
11. Top 8 MP3 Players Application and Software for Any Platform [Electronic resource]. – Access mode: <https://www.anymp4.com/player/best-mp3-player.html>.

12. Dopamine [Electronic resource]. – Access mode:
<https://dopamine.en.uptodown.com/windows>.
13. Clementine [Electronic resource]. – Access mode:
<https://www.clementine-player.org/en/>.
14. Foobar [Electronic resource]. – Access mode:
<https://www.foobar2000.org/>.
15. C# language [Electronic resource]. – Access mode:
<https://www.pluralsight.com/paths/c-10>.
16. A tour of the C# language [Electronic resource]. – Access mode:
<https://learn.microsoft.com/en-gb/dotnet/csharp/tour-of-csharp/overview>.
17. MonoDevelop Documentation [Electronic resource]. – Access mode:
<https://www.monodevelop.com/documentation/>.
18. Development Environments [Electronic resource]. – Access mode:
<https://www.mono-project.com/docs/getting-started/development-environments/>.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для підтримки процесу відтворення аудіофайлів.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення інформаційної системи відтворення аудіофайлів», затверджене наказом Національного університету «Запорізька політехніка» № 209 від 28 квітня 2025 р.

A.2 Призначення розробки

Програмний продукт призначений для підтримки процесу відтворення аудіофайлів.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення інформаційної системи для відтворення аудіофайлів забезпечує виконання таких функцій:

- підтримка можливості відтворення аудіофайлів;
- підтримка можливості вибору пристрою для відтворення аудіофайлів;
- можливість вибору характеристик звуку для відтворення;
- можливість трансляції звукових потоків іншим користувачам;
- підтримка можливості відтворення звуку, що транслюється іншими користувачами програми;
- підтримка можливості організатора звукової трансляції керування можливістю прослуховування трансляції іншими користувачами.

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення інформаційної системи для відтворення аудіофайлів повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

А.3.3 Вимоги до надійності

Програмне забезпечення інформаційної системи для відтворення аудіофайлів повинно забезпечити надійне функціонування.

А.3.4 Умови експлуатації

Для експлуатації програмного забезпечення інформаційної системи для відтворення аудіофайлів необхідна наявність персонального комп'ютера.

А.3.5 Вимоги до складу та параметрів технічний засобів

Для роботи програмного забезпечення інформаційної системи для відтворення аудіофайлів необхідно забезпечити відповідний рівень технічного та програмного оснащення. Серед апаратних вимог – наявність процесора, не із частотою не менше 500 МГц, а також обсяг оперативної пам'яті щонайменше 2 гігабайти, хоча для стабільної роботи системи бажано мати більше. Жорсткий диск має містити вільний простір у розмірі щонайменше 500 мегабайт для інсталяції основних компонентів, при цьому бажано мати додаткове місце для тимчасового зберігання аудіо-даних. Аудіосистема комп'ютера повинна включати звукову карту. Також потрібен стандартний

мережевий адаптер, який забезпечує доступ до локальної мережі для передачі аудіопотоків.

З програмного боку, система повинна функціонувати під керуванням операційної системи, наприклад, Windows або дистрибутива Linux, що підтримує звукові підсистеми типу ALSA чи PulseAudio. Для створення аудіо-трансляцій на серверній стороні необхідно встановити додаток, який підтримує передачу звукових потоків за протоколом TCP, а також мати засіб для ретрансляції звуку до інших пристроїв або комп'ютерів у мережі.

На клієнтських комп'ютерах мають бути встановлені компоненти, що дозволяють приймати TCP-трансляції для реалізації дзвінків або корпоративного аудіомовлення. При потребі користувач також може скористатися стандартним програвачем для прослуховування окремих аудіофайлів.

A.3.6 Вимоги до маркування і пакування

Програма для відтворення аудіофайлів може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення інформаційної системи для відтворення аудіофайлів».

A.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення інформаційної системи для відтворення аудіофайлів є звукові файли або звукові потоки.

Вихідними даними програмного забезпечення інформаційної системи для відтворення аудіофайлів є звуковий потік, який відтворюється на пристрої (звуковій карті) користувача.

ДОДАТОК Б

Фрагмент тексту програми

```

private void SrvBgnSRc()
{
    try
    {
        if (SrvBgnSRcFlag == false)
        {
            int bLn = 0;
            if (SrJBRCFlg)
            {
                bLn = SgndProc.GetBhpi((uint)Settings.SpgsSrvr,
Settings.BpfsSrvr, Settings.ChNmSrvr) * (int)StrVl;
            }
            else
            {
                bLn = SgndProc.GetBhpi((uint)Settings.SpgsSrvr,
Settings.BpfsSrvr, Settings.ChNmSrvr);
            }
            if (bLn > 0)
            {
                SStr = new SgndProc.Store();
                SStr.InfStr += new SgndProc. AbhsDSt(OnSgndGt);
                SStr.RcEnd += new SgndProc.
AbhsDEnded(OnServRcEnd);
                if (SStr.Begin(Settings.SInServ, Settings.SpgsSrvr,
Settings.BpfsSrvr, Settings.ChNmSrvr, sgndBLn, bLn))
                {
                    DsplySrvTranslate();
                    FormMain.DicInfo[this] = new Queue<List<byte>>();
                    JSrvRc.Begin();
                }
            }
        }
    }
    catch (Exception t)
    {
        ShowError(LbCl, t.Message);
    }
}

private void EndSDStore()
{
    try
    {
        if (ClBeginSStrFlag)
        {
            CStr.End();
            CStr.InfStr -= new SgndProc. AbhsDSt(OnSgndGt);
            CStr.RcEnd -= new SgndProc.
AbhsDEnded(OnSgndEnded);
            CStr = null;
            if (JBRCFlag)
            {
                JClRc.End();
            }
        }
    }
}

```

```

        DsplClTrnslt();
    }
}
catch (Exception t)
{
    ShowError(LbCl, t.Message);
}
}

private void SrvEndSRc()
{
    try
    {
        if (SrvBgnSRcFlag)
        {
            SStr.End();
            SStr.InfStr -= new SgndProc. AbhsDSt(OnSgndGt);
            SStr.RcEnd -= new SgndProc.
AbhsDEnded(OnSrvRcEnd);
            SStr = null;
            JSrvRc.End();
            DsplySrvTranslateEnd();
        }
    }
    catch (Exception t)
    {
        ShowError(LbCl, t.Message);
    }
}

private void OnSgndGt(Byte[] sgnd)
{
    try
    {
        lock (this)
        {
            if (ClInCnFlg)
            {
                if (Settings.ClMute == false)
                {
                    int bhpi =
SgndProc.GetBhpi((uint)Settings.SpgsKlnt, Settings.BpfsKlnt,
Settings.ChNmClnt);
                    int size = sgnd.Length / bhpi;
                    int cp = 0;
                    for (int i = 0; i < size; i++)
                    {
                        Byte[] dt = new Byte[bhpi];
                        Array.Copy(sgnd, cp, dt, 0, bhpi);
                        cp += bhpi;
                        SgndProc.RHTTP rhttp = ToRHTTP(partBytes,
Settings.BpfsKlnt, Settings.ChNmClnt);
                        if (JBRcFlag)
                        {
                            JClRc.AddSgnd(rhttp);
                        }
                        else {Byte[] rhttpBts = ToRHTPSgnd(sgnd,
Settings.BpfsKlnt, Settings.ChNmClnt);
                            trnsCl.Dsptch(ProtInfo.ToBytes(rhttpBts));
                        }
                    }
                }
            }
        }
    }
}

```

```

        }
    }
}
}
catch (Exception t)
{System.Diagnostics.Debug.WriteLine(t.Message);
}
}

private void OnSgndGt(Byte[] sgnd)
{try
    {lock (this)
        {if (SrvAciv)
            {if (frmTp)
                {if (Settings.SMt == false)
                    {int bhpi =
SgndProc.GetBhpi((uint)Settings.SpgsSrvr, Settings.BpfsSrvr,
Settings.ChNmSrvr);
                    int ncum = sgnd.Length / bhpi;
                    int cp = 0;
                    for (int i = 0; i < ncum; i++)
                    {Byte[] dt = new Byte[bhpi];
                    Array.Copy(sgnd, cp, dt, 0, bhpi);
                    cp += bhpi;
                    Queue<List<Byte>> d = FormMain.DicInfo[this];
                    if (d.Count < 10)
                    {d.Enqueue(new List<Byte>(dt));
                        }
                    }
                }
            }
        }
        catch (Exception t)
        {System.Diagnostics.Debug.WriteLine(t.Message);
        }
    }

private Byte[] ToRHTPSgnd(Byte[] sgnd, int bpfs, int chNcum)
{
    SgndProc.RHTP rhttp = ToRHTP(sgnd, bpfs, chNcum);
    Byte[] dt = rhttp.ToBytes();
    return dt;
}

private SgndProc.RHTP ToRHTP(Byte[] sgnd, int bpfs, int
chNcum)
{
    Byte[] dt = SgndProc.Utills.TransformSgnd(sgnd, bpfs,
chNcum);
    SgndProc.RHTP s = new SgndProc.RHTP();

```

```

        s.Sgnd = dt;
        s.CSFRC = CSFRC;
        s.X = ExFlg;
        s.SL = SgndProc.RHTP.MinLen;
        s.M = M;
        s.P = Offset;
        s.PT = pt;
        s.V = vers;
        s.Src = srcNcum;
        try
        {
            s.SN = Convert.ToUInt16(SN);
            SN++;
        }
        catch (Exception)
        {
            SN = 0;
        }
        return s;
    }

    private void InCnClnt()
    {try
        {if (ClInCnFlg == false)
            {if (Settings.ClAdr.Length > 0 && Settings.ClPrt> 0)
                {trnsCl = new NF.TrnsltKlnt(Settings.ClAdr,
Settings.ClPrt);
                    trnsCl.KlntInConn += new
NF.TrnsltKlnt.DgtConn(OnKlntInConn);
                    trnsCl.KlntOutCn += new
NF.TrnsltKlnt.DgtConn(OnKlntOutCn);
                    trnsCl.ErInf += new
NF.TrnsltKlnt.DgtExc(OnKlntErInf);
                    trnsCl.SgndGot += new
NF.TrnsltKlnt.DgtSgndGot(OnKlntSgndGot);
                    trnsCl.InConn();
                }
            }
        }
        catch (Exception t)
        {trnsCl = null;
            ShowError(LbCl, t.Message);
        }
    }

    private void OutCnKlnt()
    {try
        {EndSDStore();

```

```

        if (trnsCl != null)
        {trnsCl.OutCn();
            trnsCl.KlntInConn -= new
NF.TrnsltKlnt.DgtConn(OnKlntInConn);
            trnsCl.KlntOutCn -= new
NF.TrnsltKlnt.DgtConn(OnKlntOutCn);
            trnsCl.ErInf -= new
NF.TrnsltKlnt.DgtExc(OnKlntErInf);
            trnsCl.SgndGot -= new
NF.TrnsltKlnt.DgtSgndGot(OnKlntSgndGot);
            trnsCl = null;
        }
    }
    catch (Exception t)
    {ShowError(LbCl, t.Message);
    }
}

private void BeginSrvr()
{try
    {if (!SrvAciv)
        {if (Settings.SpgsSrvr.Length > 0 && Settings.Sprrt > 0)
            {trnsSrv = new NF.TranslateSrvr();
                trnsSrv.KlntInConn += new
NF.TranslateSrvr.DgtCIC(OnSCInConn);
                trnsSrv.KlntOutCn += new
NF.TranslateSrvr.DgtCOC(OnSCOutCn);
                trnsSrv.SgndGot += new
NF.TranslateSrvr.DgtSgndGot(OnSrvrSgndGot);
                trnsSrv.Begin(Settings.SpgsSrvr, Settings.Sprrt);
                if (trnsSrv.State ==
NF.TranslateSrvr.LstStats.Began)
                    {DsplySrvBegin();
                    }
                else
                    {DsplySrvEnd();
                    }
            }
        }
    }

private void EndSrvr()
{try
    {if (SrvAciv)
        {StopServThreads();
            trnsSrv.End();
            trnsSrv.KlntInConn -= new
NF.TranslateSrvr.DgtCIC(OnSCInConn);

```

```

        trnsSrv.KlntOutCn -= new
NF.TranslateSrvr.DgtCOC(OnSCOutCn);
        trnsSrv.SgndGot -= new
NF.TranslateSrvr.DgtSgndGot(OnSrvrSgndGot);
    }
    if (trnsSrv != null)
    {if (trnsSrv.State == NF.TranslateSrvr.LstStats.Began)
        {DsplySrvBegin();
        }
        else
        {DsplySrvEnd();
        }
    }
    trnsSrv = null;
}

}

private void ClSDRproduceBegin()
{if (SDRprFlag)
    {if (JC1Rp != null)
        {JbRprInit();
        JC1Rp.Begin();
        }
        if (CRprod == null)
        {CRprod = new SgndProc.Rproduce();
        CRprod.Open(Settings.ClOutSgnd, Settings.SpgsKlnt,
Settings.BpfsKlnt, Settings.ChNmClnt, (int)Settings.JBClNcum);
        }
        pbClTmr.Begin();
        }
        CbCSt.Invoke(new MethodInvoker(delegate()
        {CbCSt.Enabled = false;
        UdJC1.Enabled = false;
        PbCl.Max = (int)JC1Rp.Max;
        })));
}

private void ClSDRproduceEnd()
{if (CRprod != null)
    {CRprod.Close();
    CRprod = null;
    }
    if (JC1Rp != null)
    {JC1Rp.End();
    }
    pbClTmr.End();
    this.Invoke(new MethodInvoker(delegate()
    {CbCSt.Enabled = true;

```

```

        UdJCl.Enabled = true;
        PbCl.Value = 0;
    }));
}
}

public void Begin(string adr, int Prt)
{IPp = new IPEndPoint(IPAddress.Parse(adr), Prt);
  tcplst = new Tcplstener(IPp);
  if (tcplst == null) return;
  try
  {tcplst.Begin();
    thd = new Thread(new ThreadStart(Run));
    thd.Begin();
    this.lstSt = LstStats.Began;
  }
  catch (Exception t)
  {tcplst.End();
    this.lstSt = LstStats.Error;
    throw t;
  }
}

private void Run()
{while (true)
  {TrnsltKlnt tcl = tcplst.TrnsltKlntOK();
    SrvrThread sthd = new SrvrThread(tcl);
    sthd.SgndGot += new
SrvrThread.DgtSgndGot(OnSgndGt);
    sthd.KlntOutCn += new
SrvrThread.DgtCOC(OnKlntOutCn);
    OnKlntInConn(sthd);
    try
    {tcl.Klnt.BeginGet(sthd.ReadBuffer, 0, sthd.ReadBuffer.Length,
SocketFlags.None, sthd.Receive, tcl.Klnt);
      }
      catch (Exception t)
      {Console.WriteLine(t.Message);
      }
    }
  }
}

public int Dsptch(Byte[] sgnd)
{List<SrvrThread> tmp = new List<SrvrThread>(thdList);
  foreach (SrvrThread sv in tmp)
  {try
    {if (sgnd.Length > 0)
      {sv.Dsptch(sgnd);

```

```

        }
    }
    return thdList.Count;
}
public void End()
{try
    {if (thd != null)
        {thd.Abort();
            System.Threading.Thread.Sleep(100);
        }
        for (IEnumerator en = thdList.GetEnumerator();
en.MoveNext(); )
        {SrvrThread st = (SrvrThread)en.Current;
            st.End();
            if (KlntOutCn != null)
                {KlntOutCn(st, "KlntOutCn");
                }
            }
        if (tcplst != null)
            {tcplst.End();
                tcplst.Srvr.Close();
            }
        thdList.Clear();
        this.lstSt = LstStats.Ended;
    }
    catch (Exception)
    {this.lstSt = LstStats.Error;
    }
}
}

{public class TrnsltKlnt
    {public TrnsltKlnt(String adress, int p)
        {this.trnsSrv = adress;
            this.sprt = p;
            this.ErInf += new DgtExc(this.OnErInf);
            this.KlntInConn += new DgtConn(this.OnInConned);
            this.KlntOutCn += new DgtConn(this.OnOutCned);
        }

        public TrnsltKlnt Klnt;
        NetworkStream sgndThread;
        byte[] SgndBuff;
        String trnsSrv;
        int sprt;
        private int CnAgnDlt = 20;
        bool CnAgn = false;
    }
}

```

```

private System.Threading.Timer CnAgnTimer;

public override string GetText()
{return String.Format("{0} {1}:{2}", this.GetType(),
this.trnsSrv, this.sprrt);
}

private void CnAgnTimerInit()
{if (CnAgn)
    {if (CnAgnTimer == null)
        {if (CnAgnDlt > 0)
            {CnAgnTimer = new System.Threading.Timer(new
System.Threading.TimerCallback(OnCnAgnTimer), null, CnAgnDlt *
1000, CnAgnDlt * 1000);
            }
        }
    }
}

public void Dsptch(Byte[] bv)
{try
    {sgndThread.Write(bv, 0, bv.Length);
    if (this.SgndDsptch != null)
        {this.SgndDsptch(this, bv);
        }
    }
    catch (Exception t)
    {ErInf(this, t);
    }
}

private void BgnPrs()
{try
    {SgndBuff = new byte[1024];
    sgndThread.BeginRead(SgndBuff, 0,
SgndBuff.Length, new AsyncCallback(OnSgndGt), sgndThread);
    }
    catch (Exception t)
    {ErInf(this, t);
    }
}

private void OnSgndGt(IAsyncResult asres)
{try
    {NetworkStream ntThd = (NetworkStream)
asres.AsyncState;
    if (ntThd.CanRead)
        {int bhsz = ntThd.EndRead(asres);

```

```

        if (bhsz > 0)
        {
            if (this.SgndGot != null)
            {
                Byte[] sgnd = new byte[bhsz];
                System.Array.Copy(SgndBuff, 0, sgnd,
0, bhsz);

                this.SgndGot(this, sgnd);
            }
        }
        else
        {
            if (this.KlntOutCn != null)
            {
                this.KlntOutCn(this, "END");
            }
            if (CnAgn == false)
            {
                this.PrvOutCn();
            }
            else
            {
                this.OutCnAgn();
            }
            return;
        }
        ntThd.BeginRead(SgndBuff, 0, SgndBuff.Length, new
AsyncCallback(OnSgndGt), ntThd);
    }
}
catch (Exception t)
{
    ErInf(this, t);
}
}

public void ReNewCn()
{
    this.OutCn();
    this.InConn();
}
public void InConn()
{
    try
    {
        CnAgnTimerInit();
        Klnt = new TrnsltKlnt(this.trnsSrv, this.sprt);
        sgndThread = Klnt.RetNetwThread();
        this.BgnPrs();
        KlntInConn(this, String.Format("сервер: {0} порт:
{1}", this.trnsSrv, this.sprt));
    }
    catch (Exception t)
    {
        throw t;
    }
}

public void Png()

```

```

        {Png res = new Png();
          PngReply ans = res.Send(trnsSrv);
          if (ans.Sttgh != IPSttgh.Scs)
            {throw new Exception(String.Format("сервер {0} пінг
", trnsSrv));
              }
        }

    public void Png(Int32 interv)
    {Png res = new Png();
      PngReply ans = res.Dsptch(trnsSrv, interv);
      if (ans.Sttgh != IPSttgh.Scs)
        {throw new Exception(String.Format("сервер {0} пінг
", trnsSrv));
          }
    }

    public void OutCn()
    {PrvOutCn();
      if (CnAgnTimer != null)
        {CnAgnTimer.Dispose();
          CnAgnTimer = null;
        }
    }
    private void OutCnAgn()
    {PrvOutCn();
    }
  }
}

```

```

private void OnTransEnded()
{try
    {this.Invoke(new MethodInvoker(delegate()
    {CbIgnfow.Enabled = true;
      CbExit.Enabled = true;
      CmhS.Enabled = true;
      CjhS.Enabled = true;
      CmChNcum.Enabled = true;
      CbNcum.Enabled = true;
      CbLn.Enabled = true;
      BtnBegTrns.BackColor = SystemColors.Control;
    })));
    }
    catch (Exception t)
    {MessageBox.Show(t.Message, "Помилка");
    }
}

```

```

private void BtnBegTrnsOnClick(object Dsptcher, EventArgs e)
{try
    {{if (Extrsm.Began)
        {Extrsm.End();
        }
        else
        {int spgs = Convert.ToInt32(CmhS.SelectedItem);
        int bpfs = Convert.ToInt32(CjhS.SelectedItem);
        int chNcum =
Convert.ToInt32(CmChNcum.SelectedItem);
        int bn = Convert.ToInt32(CbNcum.SelectedItem);
        int bl = Convert.ToInt32(CbLn.SelectedItem);
        Extrsm.Begin(CbIgnfow.SelectedItem.ToString(),
CbExit.SelectedItem.ToString(), spgs, bpfs, chNcum, bn, bl);
        CbIgnfow.Enabled = false;
        CbExit.Enabled = false;
        CmhS.Enabled = false;
        CjhS.Enabled = false;
        CmChNcum.Enabled = false;
        CbNcum.Enabled = false;
        CbLn.Enabled = false;
        BtnBegTrns.BackColor = Color.Green;
        }
    }
}

```

```

SgndProc.Rproduce sPlg = new SgndProc.Rproduce();
SgndProc.Store sRc = new SgndProc.Store();
List<Byte> RecSgnd = new List<Byte>();
String Fname = "";
int Spgs = 4000;
int Bpfs = 8;
int ChNcum = 2;
bool Addgr = false;

```

```

private void BgnRc()
{try
    {if (TbFnm.Text.Length > 0)
        {if (sRc.Began == false)
            {Spgs = Convert.ToInt32(CmhS.Text);
            Bpfs = Convert.ToInt32(CjhS.Text);
            ChNcum = Convert.ToInt32(CmChNcum.Text);
            string infw =
CbIgnfow.SelectedItem.ToString();
            Fname = TbFnm.Text;
            Addgr = CbAddgr.Checked;
            int bhs = CalcBhszUsingSpgs(Spgs);

```

```

        sRc.Begin(infw, Spgs, Bpfs, ChNcum, 8,
bhs);
        RcDsplg(true);
    }
}
}

```

```
private int CalcBhszUsingSpgs(int spgs)
```

```

{int bhs = 2048;
    if (spgs >= 5000)
    { bhs = 512;
    }
    if (spgs >= 8000)
    { bhs = 1024;
    }
    if (spgs >= 11025)
    { bhs = 1024;
    }
    if (spgs >= 22050)
    { bhs = 2048;
    }
    if (spgs >= 44100)
    { bhs = 4096;
    }
    if (spgs >= 96000)
    { bhs = 8192;
    }
    return bhs;
}

```

```
private void EndRc()
```

```

{try
    {if (sRc.Began)
        {sRc.End();
        }
    }
}

```

```
private void BtnPauseOnClick(object Dsptcher, EventArgs e)
```

```

{try
    {if (sPlg.IsOnPause)
    {bool res = sPlg.EndPause();
    }
    else
    {bool res = sPlg.BeginPause();
    }
}
}

```

```

        private void FrmMainOnClose(object Dsptcher,
FormClosingEventArgs e)
        {try
            {
                sPlg.CPl -= new SgndProc.Rproduce.AbhsDEnded(OnCPl);
                sPlg.EPl -= new SgndProc.Rproduce.AbhsDEnded(OnEPl);
                sRc.InfStr -= new SgndProc.Store.AbhsDSt(OnInfStr);
                sRc.RcEnd -= new SgndProc.Store.AbhsDEnded(OnRcEnd);
                bool res = sPlg.Close();
                res = sRc.End();
            }
        }

        private void BtnRecOnClick(object Dsptcher, EventArgs e)
        {if (sRc.Began)
            {EndRc();
            }
            else
            {BgnRc();
            }
        }
    }
}

```

ДОДАТОК В
Слайди презентації

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

Програмне забезпечення інформаційної системи відтворення аудіофайлів

Виконала: Аліса УДОД
ст. групи КНТ-211

Керівник: Євгеній ГОФМАН
к.т.н., доцент

Рисунок В.1 – Слайд 1

Об'єкт, предмет та мета роботи

- **Об'єкт дослідження** – процес розробки засобів для відтворення аудіофайлів.
- **Предмет дослідження** – засоби для відтворення аудіофайлів.
- **Метою роботи** є розробка програмного забезпечення для відтворення аудіофайлів.

2

Рисунок В.2 – Слайд 2

Завдання роботи

- виконати аналіз предметної області та програмного забезпечення для відтворення аудіофайлів;
- здійснити проектування програмного забезпечення для відтворення аудіофайлів;
- створити програмного забезпечення для відтворення аудіофайлів;
- виконати тестування розробленого програмного забезпечення інформаційної системи для відтворення аудіофайлів.

3

Рисунок В.3 – Слайд 3

Порівняння існуючих аналогів

Критерій порівняння	Dopamine	Clementine	Foobar
Простота інтерфейсу	+	+—	—
Підтримка великої кількості звукових форматів	+—	+	+
Відтворення аудіо з архівів	—	+—	+
Конвертація аудіо	—	+	+
Візуалізація	+—	+	—
Підтримка потокового аудіо	—	+	+—

4

Рисунок В.4 – Слайд 4

Вимоги до програмного забезпечення

- підтримка можливості відтворення аудіофайлів;
- підтримка можливості вибору пристрою для відтворення аудіофайлів;
- можливість вибору характеристик звуку для відтворення;
- можливість трансляції звукових потоків іншим користувачам;
- підтримка можливості відтворення звуку, що транслюється іншими користувачами програми;
- підтримка можливості організатора звукової трансляції керування можливістю прослуховування трансляції іншими користувачами.

5

Рисунок В.5 – Слайд 5

Порівняння мов програмування

Критерій порівняння мов програмування	Мова програмування		
	C#	Python	Java
Наявність мультимедійних бібліотек	+	+	+–
Оптимізація роботи з потоками та фоновими задачами	+	+–	+
Підтримка асинхронного програмування	+	+	+–
Зручність для створення ПЗ для відтворення аудіофайлів	+	–	–
Простота реалізації аудіофункціоналу	+	+–	–

6

Рисунок В.6 – Слайд 6

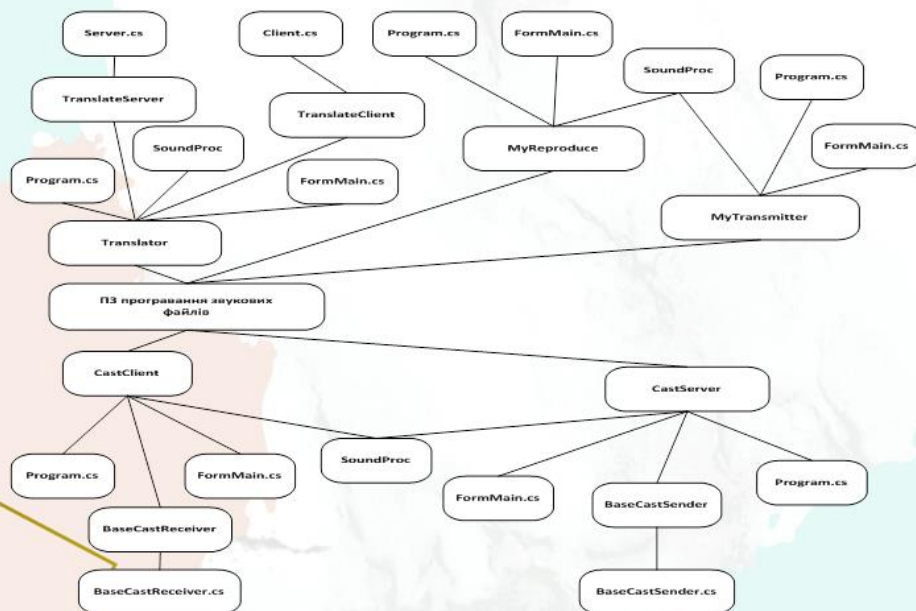
Порівняння середовищ розробки

Критерій порівняння середовищ розробки	Середовища розробки		
	MonoDevelop	Visual Studio	Rider
Підтримка роботи з мультимедіа	+	+—	+—
Простота інтерфейсу	+	+	+—
Кросплатформність	+	—	+
Швидкість запуску та роботи	+	—	+—
Ліцензія та безкоштовність	+	+—	—

7

Рисунок В.7 – Слайд 7

Структура ПЗ для відтворення аудіофайлів



8

Рисунок В.8 – Слайд 8

Схема функціонування програми

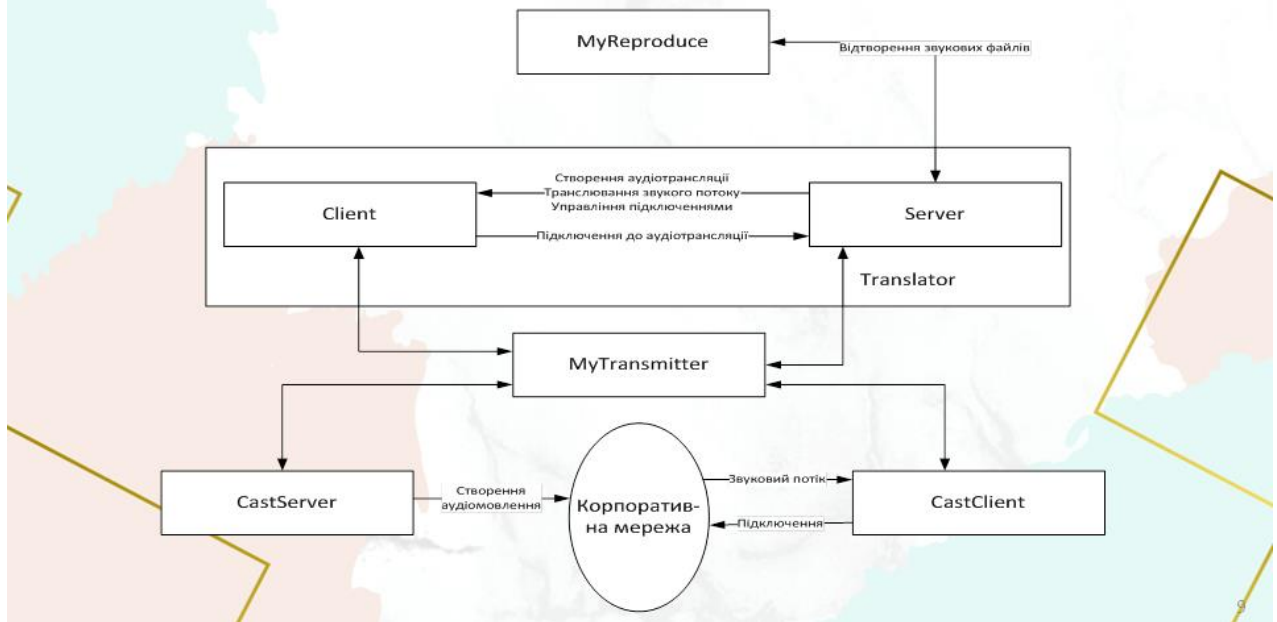


Рисунок В.9 – Слайд 9

Діаграма прецедентів програми

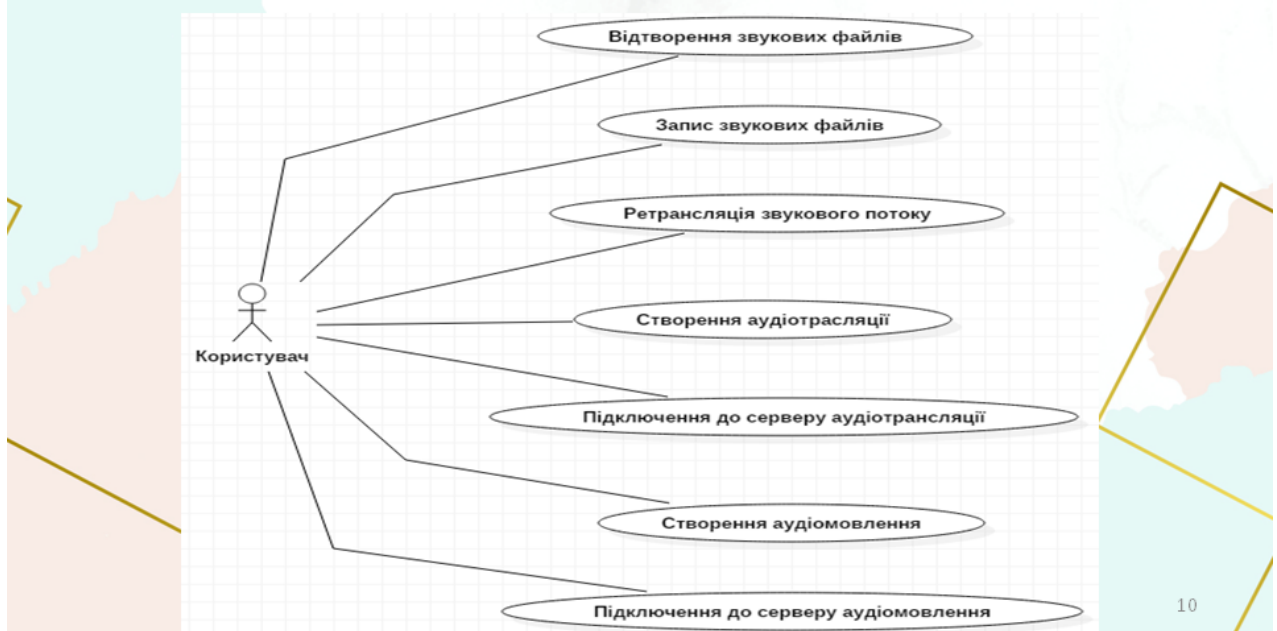
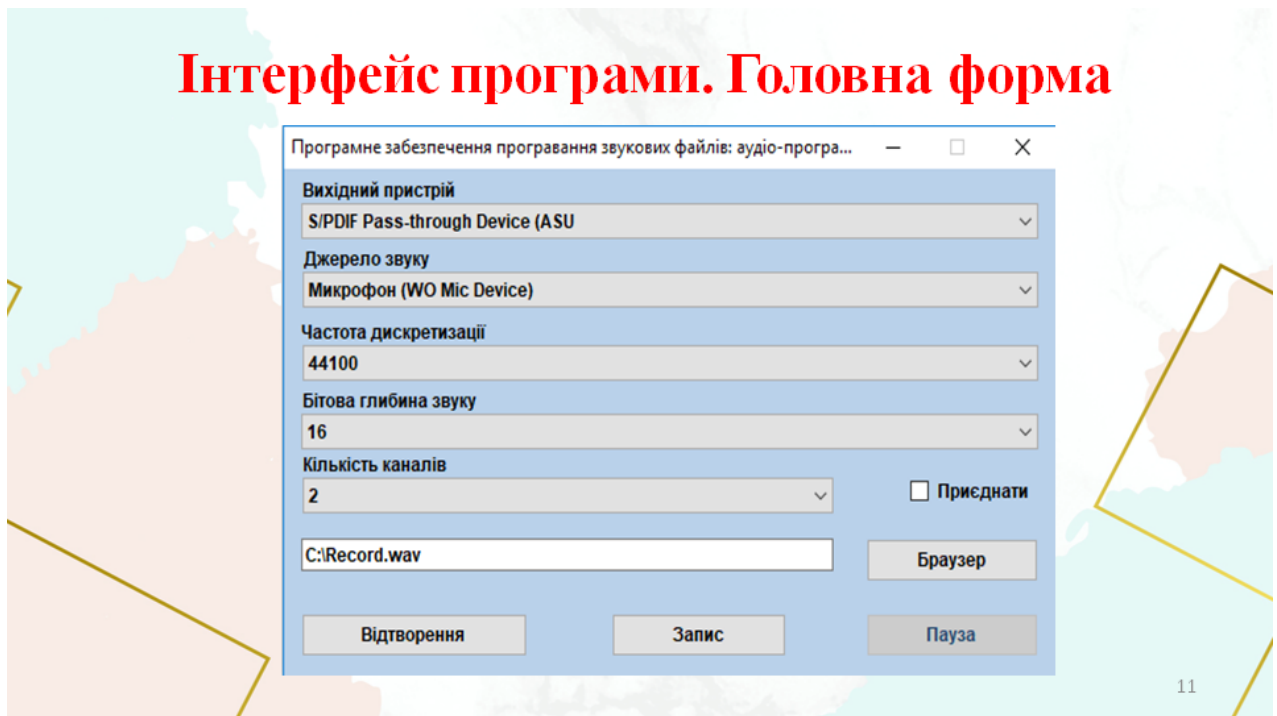


Рисунок В.10 – Слайд 10

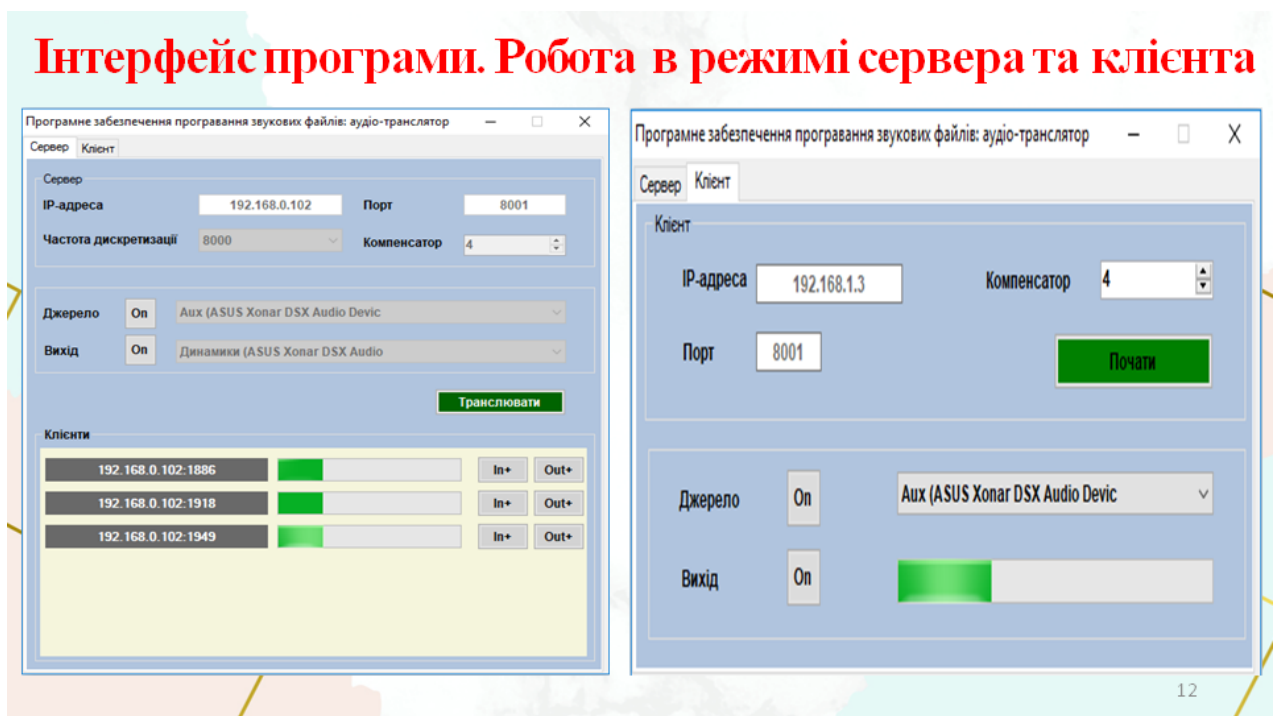
Інтерфейс програми. Головна форма



11

Рисунок В.11 – Слайд 11

Інтерфейс програми. Робота в режимі сервера та клієнта



12

Рисунок В.12 – Слайд 12

Висновки

В ході виконання кваліфікаційної роботи було проаналізовано та досліджено процес розробки засобів для відтворення аудіофайлів.

Сформульовано функціональні вимоги до програмного забезпечення інформаційної системи для відтворення аудіофайлів. Для реалізації програмного забезпечення інформаційної системи для відтворення аудіофайлів обрано мову програмування C# та середовище розробки MonoDevelop. Визначено, що у програмному забезпеченні для відтворення аудіофайлів реалізовано кілька функціональних блоків, які забезпечують як локальне відтворення, так і передачу звукових потоків через мережу. Завдяки такій модульній організації забезпечується розширюваність, підтримка різних сценаріїв використання та масштабування системи.

Практичне значення роботи полягає в тому, що розроблено програмне забезпечення для оцінювання впливу шкідливих речовин.

Усі завдання кваліфікаційної роботи повністю виконано.

13

Рисунок В.13 – Слайд 13