Міністерство освіти і науки України
Одеський національний технологічний університет
Кафедра комп'ютерної інженерії

ПОЯСНЮВАЛЬНА ЗАПИСКА ДО КВАЛІФІКАЦІЙНОЇ РОБОТИ

на тему Система передачі зображення екрана через мережу
(назва кваліфікаційної роботи згідно наказу ОНТУ)

Здобувача Іванова Ю.С.
(прізвище, ініціали)
2 курсу 543а групи

Керівники: д.т.н., проф. Артеменко С.В.
(посада, прізвище та ініціали)
ст. викл. Сіренко О.І.
(посада, прізвище та ініціали)

Консультанти:
(посада, прізвище та ініціали)
Phd, ст.викл. Богданов О.О.
(посада, прізвище та ініціали)

Кваліфікаційна робота допускається до захисту

Рішення кафедри від *05.06* 2024 р., протокол № *8*
Завідувач кафедри комп. інженерії _____ Сергій АРТЕМЕНКО
(назва кафедри) (підпис) (Ім'я ПРІЗВИЩЕ)

Одеса – 2024 рік

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерної інженерії, програмування та кіберзахисту*
Кафедра *комп'ютерної інженерії*
Ступінь вищої освіти *бакалавр*
Спеціальність *123 «Комп'ютерна інженерія»*
Освітня програма *Мережеві технології та інтернет речей*

ЗАТВЕРДЖУЮ
Зав. кафедри *комп'ютерної інженерії*
Сергій АРТЕМЕНКО
« *30* » *серпня* 2023 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

Іванова Юрія Сергійовича

1. Тема роботи *Система передачі зображення екрана через мережу*

Затверджена наказом університету від « *30* » *серпня* 2023 р., наказ № *442-03*

2 Термін здачі здобувачем закінченої роботи *28 травня 2024 р.*

3. Вихідні дані роботи

1. Середовища програмування Android Studio, IntelliJ IDEA. 2. Мова програмування Java.
3. Текстовий редактор Microsoft Word. 4. Редактор презентацій Microsoft PowerPoint.

4. Перелік питань, які потрібно розробити

1. Вступ. 2. Збір та аналіз інформації. 3. Проектування системи.
4. Розробка системи. 5. Економічні розрахунки.
6. Охорона праці. 7. Загальні висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Слайд 1. Тема роботи. Слайд 2. Об'єкт та предмет дослідження. Слайд 3. Актуальність.
Слайд 4. Аналоги систем. Слайд 5. Задача проекту. Слайд 6. Діаграма Use-Case.
Слайд 7. Діаграма дій створення серверу.
Слайд 8. Діаграма дій підключення пристрою до сервера. Слайд 9. Діаграма класів.
Слайд 10. Головне вікно застосунку. Слайд 11. Вікно відображення зображення екрану.
Слайд 12. Техніко-економічні показники проекту. Слайд 13. Висновки.

6. Консультанти по роботі, із зазначенням розділів роботи, що стосуються їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Економіка	*Phd, ст. викл. Богданов О.О.*		
Охорона праці	*д.т.н., проф. Артеменко С.В.*		
Нормоконтроль	*ст. викл. Сіренко О.І.*		

7. Дата видачі завдання *30.08.2023*

Керівники *Сергій АРТЕМЕНКО*

Олександр СІРЕНКО

Завдання прийняв до виконання *Юрій ІВАНОВ*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	*Дослідженні предметної області*	*26.10.2023*	
2.	*Дослідження існуючих аналогів*	*30.11.2023*	
3.	*Дослідження методів реалізації передачі*	*28.01.2024*	
	зображення екрану через мережу		
4.	*Проектування*	*15.02.2024*	
5.	*Розробка демонстраційної версії ПЗ*	*27.03.2024*	
6.	*Підготовка техніко-економічної частини*	*15.04.2024*	
7.	*Підготовка розділу охорони праці*	*15.04.2024*	
8.	*Оформлення пояснювальної записки*	*27.05.2024*	
9.	*Оформлення графічної частини та лістингу*	*27.05.2024*	

Здобувач-дипломник _____ *Юрій ІВАНОВ*

Керівники роботи _____ *Сергій АРТЕМЕНКО*

_____ *Олександр СІРЕНКО*

Несу відповідальність за ідентичність електронного та друкованого варіантівкваліфікаційної роботи, даю згоду на обробку персональних даних та не заперечую проти розміщення кваліфікаційної роботи на офіційних web-ресурсах ОНТУ.

Підтверджую, що в кваліфікаційній роботі відсутні порушення норм академічної доброчесності.

Здобувач-дипломник *Юрій ІВАНОВ* _____

АНОТАЦІЯ

Кваліфікаційна робота присвячена розробці системи передачі зображення екрану через мережу. Ця технологія особливо актуальна в умовах віддаленої роботи та навчання, оскільки дозволяє користувачам отримувати доступ до ресурсів на відстані, спостерігати за діями на екрані в реальному часі та взаємодіяти з віддаленими робочими столами. У першому розділі розглянуті особливості існуючих технологій передачі зображення екрану та проведено їх аналіз. У другому розділі обґрунтовано вибір засобів. У третьому розділі описано процес розробки та використання системи. У четвертому розділі проведена оцінка ефективності розробки, а також описано маркетинговий, науково-технічний, економічний, соціальний та екологічний ефект від проекту. У п'ятому розділі розглянуто питання охорони праці. Результатом роботи є функціональна застосунок здібний до передачі зображення екрану через мережу.

Ключові слова: передача зображення екрану, мережа, віддалений доступ, реальний час, застосунок.

ABSTRACT

The qualification work is devoted to the development of a screen image transfer system over the network. This technology is especially relevant in remote work and learning environments, as it allows users to access resources from a distance, observe actions on the screen in real time, and interact with remote desktops. In the first chapter, the features of the existing screen image transmission technologies are considered and their analysis is carried out. In the second section, the choice of means is substantiated. The third chapter describes the process of developing and using the system. The fourth chapter evaluates the effectiveness of the development, and also describes the marketing, scientific and technical, economic, social and environmental effects of the project. The fifth chapter deals with the issue of labor protection. The result of the work is a functional application capable of transmitting the screen image over the network.

Keywords: screen image transfer, network, remote access, real time, application.

ЗМІСТ

ВСТУП

В сучасному світі передача зображення екрана через мережу стала необхідною складовою багатьох аспектів життя, від відеоконференцій до дистанційної підтримки користувачів. Ця технологія використовується в різноманітних галузях, включаючи освіту, медицину, бізнес і розваги. Системи передачі зображення екрана через мережу дозволяють користувачам спілкуватися та співпрацювати в реальному часі, незалежно від їх географічного розташування.

Технологія передачі зображення екрана через мережу полягає в тому, що зображення, відображене на екрані одного пристрою, передається через мережу і відтворюється на екрані іншого пристрою. Це може бути реалізовано за допомогою різних протоколів і програмних рішень.

Системи передачі зображення екрана мають широкий спектр застосувань. У сфері бізнесу вони дозволяють проводити віддалені презентації, відеоконференції та спільну роботу над документами. У сфері медицини ця технологія може використовуватися для дистанційної консультації лікаря або вивчення медичних зображень. В освітній галузі системи передачі зображення екрана допомагають здійснювати дистанційне навчання та співпрацю над проектами. У розважальній індустрії вони можуть бути використані для стрімінгу відеоігор або віддаленого перегляду відео.

Метою даного дослідження є аналіз технологій передачі зображення екрана через мережу та розробка програмного забезпечення передачі зображення екрана через мережу. Для досягнення цієї мети необхідно виконати ряд завдань:

– оглянути існуючі технології передачі зображення екрана через мережу;

– проаналізувати переваги та недоліки кожної з цих технологій;

– проектування та розробка системи передачі зображення екрана через мережу.

Об'єктом дослідження є процес передачі зображення екрана через мережу, а предметом – реалізація методів передачі зображення екрану через мережу. Дослідження проводитиметься за допомогою аналізу літературних джерел, відповідних програмних засобів та експериментальних досліджень.

Отримані результати дослідження будуть корисні для розробки програмного забезпечення, фахівців у галузі мережевих технологій, а також для практичного використання в різних сферах діяльності.

Незважаючи на наявність існуючих рішень, таких як *TeamViewer, Google Remote Desktop* та інших, все ще існують проблеми, пов'язані із затримкою передачі, якістю зображення та звуку, а також сумісністю між різними пристроями та операційними системами. Ці проблеми роблять актуальною розробку нової системи передачі зображення екрану через мережу.

Практичне значення даної роботи полягає у можливості використання розробленої системи для передачі зображення екрану та звуку між пристроями, що буде корисно як для професійної діяльності, освітніх цілей, так і для повсякденного життя.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Система передачі зображення екрана через мережу є інноваційним рішенням, яке дозволяє ділитися візуальним контентом в реальному часі між різними пристроями, незалежно від їхнього фізичного розташування. Це технологія, яка забезпечує передачу відео та графічного контенту через інтернет або локальні мережі, перетворюючи традиційні методи спілкування та співпраці.

Система функціонує на принципі захоплення зображення екрана відправника та його передачі отримувачу через мережу, дозволяючи користувачам спільно працювати над документами, проводити онлайн-презентації та навчальні сесії, а також ділитися мультимедійним контентом без необхідності фізичної присутності.

Для роботи з системою достатньо мати доступ до мережі інтернет або локальної мережі, а також спеціалізоване програмне забезпечення або апаратне рішення, що підтримує передачу зображення. Використання сучасних протоколів компресії даних забезпечує високу якість зображення при мінімальних затримках, що є ключовим для ефективної співпраці в режимі реального часу.

Особливістю таких систем є можливість масштабування від простих один-до-одного сесій до мультикастингу, де одночасно можуть брати участь десятки або навіть сотні користувачів. Це робить технологію ідеальною для проведення вебінарів, дистанційного навчання та корпоративних зустрічей.

Система також підтримує додаткові функції, як-от керування віддаленим робочим столом, передачу звуку та відео високої якості, і навіть передачу сенсорних команд, що відкриває широкі можливості для інтерактивного спілкування та співпраці.

Інтеграція таких систем у корпоративне середовище або освітній процес сприяє підвищенню продуктивності, оптимізації процесів співпраці та

розширенню можливостей дистанційної роботи. Таким чином, система передачі зображення екрана через мережу є міцним кроком у майбутнє, де відстані вже не становлять перешкоди для ефективної взаємодії.

Актуальність тематики дослідження полягає в швидкому розвитку сучасних інформаційних технологій та їх значущому впливі на різні сфери життя. В контексті передачі зображення екрана через мережу, широке поширення цієї технології обумовлене зростанням потреб у віддаленій комунікації, співпраці та доступі до інформації у реальному часі.

У сучасному світі, де бізнес, освіта, медицина та розваги стають все більш глобальними, інформаційні технології відіграють ключову роль у забезпеченні ефективного спілкування та співпраці на відстані. Використання систем передачі зображення екрана через мережу дозволяє миттєво обмінюватися інформацією, проводити віддалені засідання, навчання, консультації та спільні проекти, що значно полегшує комунікацію та підвищує продуктивність праці.

Інформаційні технології в сучасній предметній галузі відіграють критичну роль у забезпеченні швидкого доступу до необхідної інформації, сприяючи підвищенню ефективності та якості послуг. Використання систем передачі зображення екрана стає невід'ємною складовою бізнес-процесів, медичної діагностики та лікування, освітнього процесу та розважальних заходів. Такі технології дозволяють збільшувати мобільність, підвищувати доступність та зручність використання інформації, що відкриває нові можливості для розвитку та модернізації різних сфер діяльності.

Отже, дослідження ролі інформаційних технологій у передачі зображення екрана через мережу має велике значення в сучасному світі, де швидкий доступ до інформації та ефективна комунікація є ключовими чинниками успіху в будь-якій галузі діяльності.

У галузі передачі зображення екрана через мережу можуть існувати деякі проблеми, які можуть виникати під час впровадження та експлуатації таких систем. Деякі з цих проблем включають:

– безпека і конфіденційність даних: передача зображення екрана через мережу може стати джерелом ризику для безпеки даних, особливо у випадку передачі конфіденційної інформації. Недостатня захищеність мережі може призвести до можливості перехоплення інформації з екрана пристрою;

– латентність та затримки: під час передачі зображення через мережу може виникнути латентність або затримка, особливо у випадку обмеженого пропускного потоку мережі або недостатньої швидкості передачі даних. Це може призвести до недоліків у реальному часі та погіршення користувальницького досвіду;

– сумісність та інтеграція: деякі програмні рішення для передачі зображення екрана можуть мати обмежену сумісність з різними операційними системами або пристроями, що може ускладнити їх інтеграцію та використання в різних середовищах;

– потреба у високій швидкості мережі: ефективна передача зображення екрана вимагає достатньо великої швидкості мережі, особливо при використанні високорозширених або відеоінтенсивних додатків. Це може стати проблемою у випадку обмежених ресурсів мережі або низької швидкості Інтернету;

– масштабованість: для великих організацій або компаній з багатьма пристроями може виникнути проблема масштабованості передачі зображення екрана через мережу. Необхідно мати ефективні рішення для керування та моніторингу багатьма одночасними з'єднаннями.

Розробка стратегій та рішень для вирішення цих проблем є ключовою для успішного впровадження та експлуатації систем передачі зображення екрана через локальну мережу.

1.2 Дослідження існуючих аналогів

Існує кілька аналогічних систем та рішень, які надають програмну підтримку для передачі зображення екрану через мережу. Деякі з них включають:

1. *TeamViewer* – програмне забезпечення для віддаленого доступу, віддаленого керування та віддаленого обслуговування комп'ютерів та інших кінцевих пристроїв, випущене у 2005 році. Інструмент підтримує *Windows*, *macOS*, певні дистрибутиви *Linux* та *Android*[1]. *TeamViewer* не вимагає реєстрації та надається безкоштовно для некомерційного використання, що забезпечило йому дуже високу популярність. *TeamViewer* - основний продукт однойменної німецької компанії з Геппінген. Вхідні та вихідні з'єднання можуть встановлюватись як через Інтернет, так і через локальні мережі. За бажанням *TeamViewer* може працювати як системна служба *Windows*, забезпечуючи доступ, що не передбачає спеціального дозволу через *TeamViewer*. Існує також портативна версія програми, яка запускається, наприклад через USB-пристрій і не вимагає установки [2].З'єднання встановлюється за допомогою автоматично створених унікальних ідентифікаторів та паролів. Перед кожним підключенням мережеві сервери *TeamViewer* перевіряють дійсність ідентифікаторів двох кінцевих точок. Безпека підвищується при скануванні відбитка пальця, що дозволяє користувачам надати додаткові докази для ідентифікації віддаленого пристрою. Паролі захищені від атак методом підбору, зокрема завдяки експоненційному збільшенню часу очікування між спробами з'єднання. *TeamViewer* надає додаткові функції безпеки, такі як двофакторна автентифікація та списки доступу.

2. *RustDesk* (рис. 1.2) — це програмне забезпечення для віддаленого доступу та дистанційного керування, в основному написане на *Rust*, яке дозволяє віддалено обслуговувати комп’ютери та інші пристрої.[3] Клієнт *RustDesk* доступний для таких операційних систем, як *Microsoft Windows, Apple macOS, Apple iOS, Android* і *Linux*. *RustDesk* прагне бути альтернативою з відкритим кодом для програмного забезпечення для віддаленого робочого столу, такого як *TeamViewer* або *AnyDesk*.[4]

3. *Parsec*— це власна програма для віддаленого робочого столу, яка в основному використовується для ігор через потокове відео. Використовуючи *Parsec*, користувач може транслювати відеоігри через підключення до

Інтернету, дозволяючи запускати гру на одному комп'ютері, але грати в неї віддалено через інший пристрій. Незважаючи на те, що його основним фокусом є ігри, *Parsec* також можна використовувати як програмне забезпечення для спільного використання робочого столу з низькою затримкою. Клієнт *Parsec* доступний у більшості сучасних операційних систем, включаючи *Windows, macOS, Android, Raspberry Pi 3* і *Linux*.[5] *Parsec* також надає платну версію «*Parsec for Teams*» з додатковими функціями для художників і розробників, такими як додаткові інструменти адміністрування, краща точність кольорів і можливість транслювати кілька екранів одночасно. *Parsec Warp* додає додаткові параметри, візуальні покращення та більше елементів керування. Режим 4:4:4 робить кольори різкішими та чіткішими. Додано підтримку пера та планшета, а також можливість мати додаткові дисплеї.[6] У 2023 році *Parsec* закрив свою аркадну систему.

4. *Chrome Remote Desktop* — це програмне забезпечення для віддаленого робочого столу, розроблене компанією *Google*, яке дозволяє користувачеві віддалено керувати робочим столом іншого комп'ютера за допомогою власного протоколу, також розробленого компанією *Google*, який називається *Chromoting*.[7] Протокол передає події клавіатури та миші від клієнта до сервера, ретранслюючи оновлення графічного екрана назад в іншому напрямку через комп'ютерну мережу. Таким чином, ця функція складається з серверного компонента для головного комп'ютера та клієнтського компонента на комп'ютері, який отримує доступ до віддаленого сервера. *Chrome Remote Desktop* використовує унікальний протокол на відміну від звичайного протоколу *Remote Desktop* (розробленого *Microsoft*[8]).

5. *AnyDesk* — це програма для віддаленого робочого столу, яку розповсюджує *AnyDesk Software GmbH*. Пропрієтарне програмне забезпечення забезпечує незалежний від платформи віддалений доступ до персональних комп'ютерів та інших пристроїв, на яких працює головна програма.[9] Він пропонує дистанційне керування, передачу файлів і функції *VPN*. *AnyDesk*

часто використовується для шахрайства з технічною підтримкою та інших шахрайств із віддаленим доступом.[10]

Таблиця 1.1

Порівняльна характеристика існуючих аналогів

Особливості	*TeamViewer*	*RustDesk*	*Parsec*	*Chrome Remote Desktop*	*AnyDesk*
Платформи	*Windows*, *macOS*, *Linux*, *Android*, *iOS*	*Windows*, *macOS*, *Linux*	*Windows*, *macOS*, *Linux*	*Windows*, *macOS*, *Linux*, *Android*, *iOS*	*Windows*, *macOS*, *Linux*, *Android*, *iOS*
Безпека	Шифрування *end-to-end*, двофакторна аутентифікація	Шифрування, *TLS*	*AES*-256 шифрування	Шифрування трафіку	Шифрування, *TLS*
Функції	Доступ до віддаленого робочого столу, передача файлів, відеоконференції	Віддалений робочий стіл, передача файлів	Віддалений робочий стіл, стрімінг відео, графіка	Доступ до віддаленого робочого столу	Віддалений робочий стіл, передача файлів

Продовження таблиці 1.1

Особливості	*TeamViewer*	*RustDesk*	*Parsec*	*Chrome Remote Desktop*	*AnyDesk*
Відмінності	Комерційна ліцензія, інтеграція зі *Slack* та іншими сервісами	Відкритий код, безкоштовний	Гравець гри, стрімінг, розширені графічні можливості	Інтегровано з *Google*-акаунтом, безкоштовно	Швидкий доступ, оптимізовано для низької затримки
Ціна	Безкоштовно для особистого використання, комерційні тарифи	Безкоштовно, преміум плани	Безкоштовно, преміум плани	Безкоштовно	Безкоштовно для особистого використання, комерційні тарифи

1.3 Мережеві технології передачі зображення екрану

Серед мережевих технологій передачі зображення екрану можна виділити деякі з них:

1. *AirPlay* - власний комунікаційний протокол бездротового зв'язку, розроблений *Apple Inc.*, що дозволяє передавати аудіо, відео, вміст екрану та фотографії (разом із відповідними метаданими) між пристроями. Спочатку *AirPlay* було реалізовано лише у власному програмному забезпеченні та пристроях *Apple*. Перша реалізація називалася *AirTunes* й підтримувала лише передачу звуку. Пізніше *Apple* ліцензувала стек протоколів *AirPlay* й для інших виробників, що створюють продукти, сумісні з пристроями *Apple*. [11] *AirPlay*

дозволяє передачу з комп'ютерів на *macOS* та *Windows* за допомогою програми *iTunes*, а також з пристроїв під керуванням *iOS*, – таких як *iPhone, iPod* та *iPad* з *iOS* 4.2 або новішої версії. Пристрої-передавачі надсилати *AirPlay* через *Wi*-Fi або *Ethernet*. Починаючи з *OS X Mountain Lion AirPlay* підтримує дзеркальне відображення дисплея через на системах, що побудовані на базі процесорів *Intel Core* 2-го покоління та новіших. [12]

2. *Google Cast* — це власний протокол, розроблений *Google* для відтворення аудіовізуального вмісту, що транслюється в Інтернеті, на сумісному споживчому пристрої. Протокол використовується для ініціювання та керування відтворенням вмісту на цифрових медіаплеєрах, телевізорах високої чіткості та домашніх аудіосистемах за допомогою мобільного пристрою, персонального комп'ютера чи розумної колонки. Протокол був вперше запущений 24 липня 2013 року для підтримки програвача *Google Chromecast* першого покоління. [13]. *Google Cast SDK* був випущений 3 лютого 2014 року, дозволяючи третім особам модифікувати своє програмне забезпечення для підтримки протоколу. [14] Станом на травень 2015 року було доступно понад 20 000 додатків із підтримкою *Google Cast*. Відтоді підтримку *Google Cast* було інтегровано в інші пристрої, такі як *Nexus Player* та інші пристрої *Android TV* (наприклад, телевізори), а також звукові панелі. , колонки та новіші моделі *Chromecast*. Споживчі пристрої, які нативно підтримують протокол, продаються як вбудовані *Chromecast*. [15]

3. *Miracast* — це стандарт бездротового зв'язку, створений *Wi-Fi Alliance*, призначений для передачі відео та звуку безпосередньо з пристроїв (наприклад, ноутбуків, планшетів або смартфонів) на приймачі (наприклад, телевізори, монітори або проектори). Його можна приблизно описати як «HDMI через *Wi-Fi*», замінюючи кабелі на користь бездротового зв'язку.[16] Протокол використовується в багатьох пристроях і використовується або має різні назви різними виробниками, зокрема *Smart View*[17] і *AllShare Cast* (від *Samsung*), *SmartShare* (від *LG*), віддзеркалення екрана (від *Sony*), *Cast* (у *Windows* 11), бездротовий дисплей і трансляція екрана.[18] *Miracast* базується

на одноранговому стандарті *Wi-Fi Direct*. Він дозволяє надсилати *HD*-відео до 1080p (кодек *H.264*) і об'ємний звук 5.1. [19] З'єднання створюється через *WPS* і тому захищене *WPA2*. *IPv4* використовується на рівні Інтернету. На транспортному рівні використовуються *TCP* або *UDP*. На прикладному рівні потік ініціюється та контролюється через *RTSP*, *RTP* для передачі даних.[20]

4. *Remote Desktop Protocol* (RDP) — це власний протокол, розроблений корпорацією Microsoft, який надає користувачеві графічний інтерфейс для підключення до іншого комп'ютера через мережеве з'єднання.[21] Для цього користувач використовує клієнтське програмне забезпечення *RDP*, тоді як інший комп'ютер має запустити програмне забезпечення сервера *RDP*. Існує кілька клієнтів для більшості версій *Microsoft Windows* (включно з *Windows Mobile*, але підтримка припинилася), *Linux* (наприклад, *Remmina*), *Unix, macOS, iOS, Android* та інших операційних систем. Сервери *RDP* вбудовані в серверні та професійні випуски операційних систем Windows, але не в домашні випуски; також існує сервер *RDP* для *Unix* і *OS X* (наприклад, *xrdp*). За замовчуванням сервер прослуховує *TCP*-порт 3389[22] і *UDP*-порт 3389.[23] Протокол є розширенням протоколу спільного використання додатків *ITU-T T.128*. *Microsoft* оприлюднює деякі специфікації на своєму веб-сайті.[24]

5. *Virtual Network Computing* (*VNC*) — це графічна система спільного використання робочого столу, яка використовує протокол *Remote Frame Buffer* (*RFB*) для віддаленого керування іншим комп'ютером. Він передає дані клавіатури та миші з одного комп'ютера на інший, ретранслюючи оновлення графічного екрана через мережу.[25] *VNC* не залежить від платформи – є клієнти та сервери для багатьох операційних систем на основі *GUI* та для *Java*. Кілька клієнтів можуть підключатися до сервера *VNC* одночасно. Популярні способи використання цієї технології включають віддалену технічну підтримку та доступ до файлів на робочому комп'ютері з домашнього комп'ютера або навпаки. *VNC* спочатку був розроблений у дослідницькій лабораторії *Olivetti & Oracle* у Кембриджі, Велика Британія. Оригінальний вихідний код VNC і багато сучасних похідних є відкритими під ліцензією *GNU General Public*

License. Існує кілька варіантів *VNC*, які пропонують власну функціональність; наприклад, деякі з них оптимізовані для *Microsoft Windows* або пропонують передачу файлів (не є частиною власне *VNC*) тощо. Багато з них сумісні (без додаткових функцій) із власне *VNC* у тому сенсі, що програма перегляду одного варіанту може підключитися до сервера іншого ; інші базуються на коді *VNC*, але не сумісні зі стандартним *VNC*.[26]

1.4 Мережеві протоколи

Сетевые протоколы играют ключевую роль в передаче данных между устройствами, обеспечивая стандартизированные методы коммуникации. К числу таких протоколов относятся:

1. *Transmission Control Protocol* (*TCP*) — разом із протоколом *IP* є стрижневим протоколом Інтернету, який дав назву моделі *TCP/IP*. Протокол призначений для керування передаванням даних у комп'ютерних мережах, працює на транспортному рівні моделі *OSI*.[27] *TCP* отримує потоки даних від протоколів верхніх рівнів *OSI*-моделі, початковим джерелом яких є протоколи прикладного рівня, такі як *HTTP*, *FTP* та інші. Кожний протокол верхнього рівня має свій визначений *TCP*-порт. *TCP* розбиває конкретний потік даних на порції, та додає до кожної з них заголовок з номером послідовності. Отримані таким чином порції даних традиційно називаються *TCP*-сегментами. Далі кожний сегмент інкапсулюється в *IP*-пакет і передається через *IP*-протокол до хоста-отримувача. [28] Після надходження *IP*-пакету до хоста-отримувача перевіряється коректність отриманих даних у *TCP*-сегменті, методом перерахування контрольної суми, та переконується, що попередні сегменти даних також були успішно отримані. Після чого хост-отримувач надсилає запит до хоста-відправника про нову, або повторне передавання порції даних, що одночасно є підтвердженням того, що всі сегменти з номерами послідовності, меншими ніж номер нового запиту, були успішно отримані. Поняття порт (*port*) є ключовим у протоколі *TCP*. Порт з точки зору операційних систем називається

сокетом (*socket*) процесу. Іншими словами це програмний інтерфейс для забезпечення обміну даними між процесами. Порти поділяються на категорії:

– загальновідомі (*well-known*), діапазон номерів 0 — 1023.[29] Як правило це порти, які використовують протоколи прикладного рівня *TCP/IP* стеку затверджені *IETF*, згідно з принципами відкритого стандарту. Наприклад: *HTTP* порт −80, *HTTPS* — 443, *SMTP* — 25, *Telnet* — 23, *SSH* — 22;

– зареєстровані (*registered*), діапазон номерів 1024 — 49151. За задумом, ці порти призначаються для протоколів різних виробників програмного забезпечення, які мають їх зареєструвати в *IANA*. Наприклад ігровий сервіс *Xbox Live* використовує зареєстрований порт 3074.[30] На практиці багато портів у цьому діапазоні використовуються без офіційної реєстрації;

– динамічні (приватні, ефемерні, *dynamic*, *private*, *ephemeral*), діапазон номерів 49152-65535. *IANA* не реєструє ці порти. Використовуються, як правило хостами-клієнтами, для встановлення *TCP*-з'єднань з хостами-серверами. Як у прикладі вище, коли клієнт з *IP*-адресою 192.168.1.31 встановив 2 з'єднання зі сервером 198.35.26.96, на стороні клієнта використовуються динамічні порти 54132 та 54138, а на стороні сервера загальновідомі порти 443 (протокол *HTTPS*) та 22 (протокол *SSH*).

Завданням протоколу, як випливає з його назви («протокол керування пересиланням»), є контроль надійного передавання даних між хостами, для забезпечення цього між хостами встановлюється логічне з'єднання, яке зветься *TCP*-сесією.[31] Протокол *TCP* працює у форматі архітектури клієнт-сервер. Хост який надсилає запит на отримання сервісу є клієнтом, той хто відповідає на запит зветься сервером. Хости зв'язуються один з одним за *TCP*-портами, на стороні клієнта це, як правило, динамічний порт, а на стороні сервера це загальновідомий або зареєстрований порт, номер якого відповідає протоколу прикладного рівня. Номери портів та значення інших параметрів заносяться

до заголовка *TCP*-сегменту. Тут під терміном сервер (*server*) розуміють комп'ютер під управлінням операційної системи, який має доступ до *IP*-мережі та надає послуги одного чи декількох сервісів прикладного рівня. В свою чергу на сервері-комп'ютері встановлені спеціальні сервер-програми, які забезпечують роботу протоколів прикладного рівня. Таким чином, якщо це вебсервер, то на ньому мусить бути встановлена одна із таких програм, як наприклад: *Apache HTTP Server*. На практиці сервер-комп'ютер надає послуги відразу декількох сервісів, тобто на ньому може бути встановлено більше однієї сервер-програми, що забезпечують роботу декількох протоколів прикладного рівня. Наприклад сервер-комп'ютер може бути одночасно вебсервером і *FTP*-сервером та забезпечувати роботу протоколів *HTTP, HTTPS* та *FTP*. Важливо розуміти, що протягом *TCP*-сесії дані надсилаються в обох напрямках, як від сервера до клієнта так і від клієнта до сервера, тобто створюються два потоки даних. Причому не завжди більший потік даних прямує від сервера до клієнта. Кожна окрема сесія роботи протоколу *TCP* може бути поділена на три фази:

– встановлення з'єднання;

– передавання даних;

– закінчення з'єднання.

2. *User Datagram Protocol*. У комп’ютерних мережах протокол дейтаграм користувача (*UDP*) є одним із основних протоколів зв’язку набору протоколів Інтернету, який використовується для надсилання повідомлень (транспортованих у вигляді дейтаграм у пакетах) на інші хости в мережі Інтернет-протоколу (*IP*). У межах *IP*-мережі *UDP* не потребує попереднього зв’язку для налаштування каналів зв’язку чи шляхів передачі даних. *UDP* використовує просту модель зв’язку без з’єднання з мінімумом механізмів протоколу. *UDP* надає контрольні суми для цілісності даних і номери портів для адресації різних функцій джерела та призначення дейтаграми. Він не має діалогів рукостискання і, таким чином, наражає програму користувача на будь-яку ненадійність основної мережі; немає гарантії доставки, замовлення або захисту дублікатів. Якщо засоби виправлення помилок необхідні на рівні мережевого

інтерфейсу, програма замість цього може використовувати протокол керування передачею (*TCP*) або протокол керування передачею потоку (*SCTP*), які призначені для цієї мети. *UDP* підходить для цілей, де перевірка та виправлення помилок або не потрібні, або виконуються в програмі; *UDP* дозволяє уникнути накладних витрат на таку обробку в стеку протоколів. Програми, чутливі до часу, часто використовують *UDP*, оскільки відкидання пакетів краще, ніж очікування пакетів, затриманих через повторну передачу, що може бути недоступним у системі реального часу.[32] *UDP* — це простий протокол транспортного рівня, орієнтований на повідомлення, задокументований у *RFC* 768. Хоча *UDP* забезпечує перевірку цілісності (через контрольну суму) заголовка та корисного навантаження[33], він не надає жодних гарантій для протоколу верхнього рівня щодо доставки повідомлень і *UDP*. рівень не зберігає стан повідомлень *UDP* після надсилання. З цієї причини *UDP* іноді називають *Unreliable Datagram Protocol*.[34] Якщо потрібна надійність передачі, це має бути реалізовано в програмі користувача. Ряд атрибутів *UDP* робить його особливо придатним для певних програм:

– він орієнтований на транзакції, підходить для простих протоколів запитів і відповідей, таких як система доменних імен або мережевий протокол часу;

– він надає дейтаграми, придатні для моделювання інших протоколів, таких як *IP*-тунелювання або віддалений виклик процедури та мережевої файлової системи;

– він простий, підходить для завантаження або інших цілей без повного стеку протоколів, таких як *DHCP* і *Trivial File Transfer Protoco*;

– він не має статусу, підходить для дуже великої кількості клієнтів, наприклад у додатках потокового медіа, як-от *IPTV*;

– відсутність затримок повторної передачі робить його придатним для додатків у реальному часі, таких як голос через *IP*, онлайн-ігри та багато протоколів, які використовують протокол потокового передавання в реальному часі;

– оскільки він підтримує багатоадресну передачу, він підходить для широкомовної інформації, наприклад для багатьох видів виявлення служб і спільної інформації, такої як протокол точного часу та протокол інформації про маршрутизацію.

Програми можуть використовувати дейтаграмні сокети для встановлення зв'язку хост-хост. Програма прив'язує сокет до своєї кінцевої точки передачі даних, яка є комбінацією *IP*-адреси та порту. Таким чином *UDP* забезпечує мультиплексування додатків. Порт — це структура програмного забезпечення, яка ідентифікується за номером порту, 16-бітним цілим значенням, що дозволяє використовувати номери портів від 0 до 65535. Порт 0 зарезервований, але є допустимим значенням вихідного порту, якщо процес надсилання не очікує повідомлень у відповідь. Управління розподілення номерів в Інтернеті (*IANA*) розділило номери портів на три діапазони. Номери портів від 0 до 1023 використовуються для звичайних, добре відомих служб. В *Unix*-подібних операційних системах для використання одного з цих портів потрібен дозвіл суперкористувача. Номери портів від 1024 до 49151 є зареєстрованими портами, які використовуються для служб, зареєстрованих *IANA*. Порти з 49152 по 65535 є динамічними портами, які офіційно не призначені для жодної конкретної служби та можуть використовуватися для будь-яких цілей. Вони також можуть використовуватися як ефемерні порти, які програмне забезпечення, що працює на хості, може використовувати для динамічного створення кінцевих точок зв'язку за потреби.[35]

Порівняємо протоколи:

1. *Transmission Control Protocol* — це протокол, орієнтований на з'єднання, і для встановлення наскрізного зв'язку потрібне встановлення зв'язку. Після встановлення з'єднання дані користувача можуть надсилатися двонаправлено через з'єднання. Має такі характеристики:

– надійність – *TCP* керує підтвердженням повідомлень, повторною передачею та тайм-аутами. Робиться кілька спроб доставити повідомлення. Якщо дані будуть втрачені в дорозі, вони будуть надіслані

повторно. У *TCP* відсутні відсутні дані, або, у разі кількох тайм-аутів, з’єднання розривається;

– упорядковано – якщо два повідомлення надсилаються через з’єднання послідовно, перше повідомлення першим надійде до програми-одержувача. Коли сегменти даних надходять у неправильному порядку, *TCP* буферизує дані, які не відповідають порядку, доки всі дані не будуть правильно впорядковані та доставлені до програми;

– важка вага – *TCP* вимагає трьох пакетів для встановлення з’єднання через сокет, перш ніж можна буде надіслати будь-які дані користувача. *TCP* забезпечує надійність і контроль перевантаження;

– потокова передача – дані зчитуються як потік байтів, жодні ознаки розрізнення не передаються до меж сигнального повідомлення (сегмента).

2. *User Datagram Protocol* — це простіший протокол без з’єднання на основі повідомлень. Протоколи без підключення не встановлюють виділене наскрізне з’єднання. Зв'язок досягається шляхом передачі інформації в одному напрямку від джерела до пункту призначення без перевірки готовності або стану приймача. Має такі характеристики:

– ненадійний – коли надсилається повідомлення *UDP*, неможливо знати, чи воно досягне місця призначення; він міг загубитися в дорозі. Немає поняття підтвердження, повторної передачі чи тайм-ауту;

– не впорядковано – якщо два повідомлення надіслано одному одержувачу, порядок їх надходження не може бути гарантований;

– легкість – немає впорядкування повідомлень, відстеження з’єднань тощо. Це дуже простий транспортний рівень, розроблений поверх *IP*;

– дейтаграми – пакети надсилаються окремо та перевіряються на цілісність після надходження. Пакети мають чіткі межі, які враховуються при отриманні; операція читання в гнізді приймача дасть повне повідомлення, як воно було надіслано спочатку;

– немає контролю над перевантаженням – сам *UDP* не уникає перевантаження. Заходи контролю перевантаження повинні бути реалізовані на рівні додатків або в мережі;

– широкомовні передачі – через відсутність з'єднання *UDP* може транслювати – надіслані пакети можуть бути адресовані таким чином, щоб вони були прийняті всіма пристроями в підмережі;

– *Multicast* – підтримується багатоадресний режим роботи, за допомогою якого один пакет дейтаграми може автоматично маршрутизуватися без дублювання до групи абонентів.

1.5 Розробка технічного завдання

На основі проведеного аналізу предметної галузі та дослідження існуючих аналогів можна сформулювати наступне технічне завдання:

Мета проекту: розробка «Системи передачі зображення екрана через мережу», яка дозволить користувачам ділитися візуальним контентом в реальному часі між різними пристроями.

Основні вимоги до системи:

– система повинна забезпечувати передачу відео та графічного контенту через інтернет або локальні мережі;
– система повинна працювати на принципі захоплення зображення екрана відправника та його передачі отримувачу через мережу;
– система повинна підтримувати роботу з документами, проведення онлайн-презентацій та навчальних сесій, а також ділитися мультимедійним контентом;
– система повинна мати високу якість зображення при мінімальних затримках;
– система повинна бути сумісна з різними операційними системами та пристроями.

Вимоги до безпеки:

– система повинна мати надійні механізми захисту для забезпечення безпеки та конфіденційності даних.

Вимоги до продуктивності:

– система повинна мати високу продуктивність та забезпечувати швидкий доступ до інформації;
– система повинна мати низьку латентність та затримку;
– система повинна мати високу швидкість передачі даних.

Вимоги до сумісності:

– система повинна бути сумісна з різними операційними системами та пристроями;
– система повинна мати можливість інтеграції з різними середовищами.

Вимоги до масштабованості:

– система повинна мати можливість масштабування для великих організацій або компаній з багатьма пристроями;
– система повинна мати ефективні рішення для керування та моніторингу багатьма одночасними з’єднаннями.

Вимоги до інтерфейсу користувача:

– інтерфейс користувача повинен бути інтуїтивно зрозумілим та простим у використанні;
– інтерфейс користувача повинен мати високу якість відображення та підтримувати високу роздільну здатність.

Вимоги до документації:

– повинна бути надана детальна документація щодо використання системи;
– документація повинна включати інструкції з встановлення, налаштування та використання системи.

Висновок до першого розділу

Аналіз предметної області та дослідження існуючих аналогів дозволили виявити ключові характеристики та потенційні можливості систем передачі зображення екрана через мережу. Ці системи відіграють важливу роль у сучасному світі, де швидкий доступ до інформації та ефективна комунікація є ключовими чинниками успіху в різних галузях діяльності.

Системи передачі зображення екрана через мережу відкривають нові можливості для розвитку та модернізації різних сфер діяльності, включаючи бізнес, освіту, медицину та розваги. Вони дозволяють збільшувати мобільність, підвищувати доступність та зручність використання інформації, що сприяє підвищенню продуктивності та оптимізації процесів співпраці.

Однак, не дивлячись на значні переваги, існують деякі проблеми, які можуть виникнути під час впровадження та експлуатації таких систем. Це включає в себе питання безпеки та конфіденційності даних, латентності та затримок, сумісності та інтеграції, потреби у високій швидкості мережі та масштабованості. Дослідження ефективних стратегій та рішень для вирішення цих проблем є важливою для успішного впровадження та експлуатації систем передачі зображення екрана через мережу.

Таким чином, враховуючи отримані результати, можна зробити висновок, що системи передачі зображення екрана через мережу є потужним інструментом, який може значно поліпшити процеси комунікації та співпраці в різних галузях. Однак, для досягнення оптимальних результатів, важливо враховувати потенційні виклики та дослідити ефективні стратегії їх вирішення.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ДОДАТКУ

2.1 Визначення функцій системи

Основними функціями системи є підключення пристроїв між собою завдяки мережевому сокету, передача та вивід зображення екранів через мережу.

Додатковими функціями є ввід та вибір *IP* адресів пристроїв, зупинка трансляції та прийняття трансляції.

2.1.1 Діаграма *Use-Case*

Use-Case - це техніка для захоплення, моделювання та визначення вимог до системи. Випадок використання відповідає набору дій, які система може виконувати у взаємодії зі своїми суб'єктами, і які дають видимий результат, який сприяє досягненню її цілей. Актори представляють роль, яку люди-користувачі або інші системи відіграють у взаємодії.

На рисунку 2.1 показано, які дії може робити користувач при роботі з застосунком.

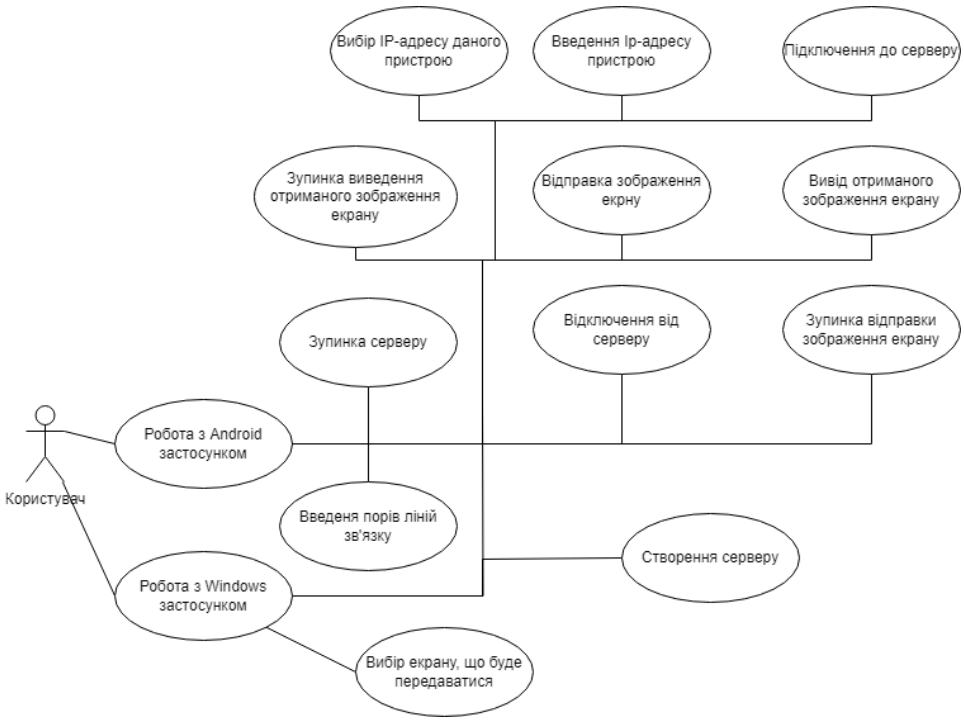


Рис. 2.1 - Діаграма *Use-Case*

У випадку вибору «Вибір адресу даного пристрою» користувач обирає *IP*-адресу з випадаючого списку для вказання адреси серверу, до якого будуть підключатися пристрої.

У випадку вибору «Введення *IP*-адресу пристрою» користувач вводить *IP*-адресу пристрою, в якому створено сервер, для підключення до нього.

У випадку вибору «Введення портів ліній зв'язку» користувач вводить порти ліній зв'язку для виділення сервером ліній передачі даних.

У випадку вибору «Створення серверу» після вводу портів ліній зв'язку та вибору *IP*-адресу серверу, при натисканні кнопки пристрій створює сервер на вказаній *IP*-адресі та з вказаними лініями зв'язку, який чекає поки буде здійснена спроба підключення до серверу або зупинка цього серверу.

У випадку вибору «Підключення до серверу» після вводу портів ліній зв'язку та *IP*-адресу пристрою, в якому створено сервер, при натисканні кнопки пристрій з'єднується з сервером та надає можливість передавати зображення екрану по створеним лініям зв'язку.

У випадку вибору «Зупинка серверу» при натисканні кнопки створений сервер розриває з'єднання з іншим пристроєм та видаляється.

У випадку вибору «Відключення від серверу» при натисканні кнопки розривається з'єднання з сервером.

У випадку вибору «Відправка зображення екрану» після підключення пристрою до серверу, при натисканні кнопки починається захват екрану та відправка зображення екрану по лініям зв'язку.

У випадку вибору «Зупинка відправки зображення екрану» при натисканні на кнопку захват та відправка зображення екрану зупиняється.

У випадку вибору «Вивід отриманого зображення екрану» після підключення пристрою до серверу, при натисканні кнопки починається прийняття зображення екрану з ліній зв'язку та відображення його в спеціальному полі.

У випадку вибору «Зупинка виводу отриманого зображення екрану» при натисканні на кнопку прийняття та відображення зображення екрану зупиняється.

У випадку вибору «Вибір екрану, що передається» так, як *Windows* пристрій може бути підключений до декількох екранів, користувач може обрати екран, що буде передаватися, з випадаючого списку.

2.1.2 Діаграма класів

Система мого проекту складається з багатьох класів і, так як система пишеться на мові *Java* і вона повинна працювати на декількох платформах, деякі класи працюють на *Android* пристроях так і на *Windows* з невеликими переробками. Такими класами э «*SocketTask*», цей клас відповідає за з'єднання пристроїв між собою та за доступ до ліній передачі даних, та «*DeviceInfo*», цей клас відповідає за збереження та передачу даних об вибраних адресах та лініях зв'язку.

В *Android* системі «*MainActivity*» являється головним класом та вікном, відповідає за передачу даних об вибраних адресах, початку з'єднання з пристроєм, перехід між вікнами, захват та передачу зображення екрана. Працює з файлом *activity_main.xml*, що описує те як повинен виглядати інтерфейс цього вікна.

Клас «*SettingsActivity*» працює з файлом *activity_settings.xml*, що описує те як повинен виглядати інтерфейс цього вікна, відповідає за передачу даних об вибраних портів ліній зв'язку.

Клас «*SurfaceActivity*» працює з файлом *activity_surface.xml*, що описує те як повинен виглядати інтерфейс цього вікна, відповідає за прийняття та відображення зображення екрану.

«*MediaService*» допоміжний клас, що відповідає за можливість захвату екрану та показ повідомлення що ведеться захват екрану.

«*NotificationButton*» допоміжний клас відповідає за функціонал кнопки в повідомлені та завершує захват та трансляцію зображення.

Більш детально взаємозв’язок класів можна побачити в діаграмі класів (рис 2.2)

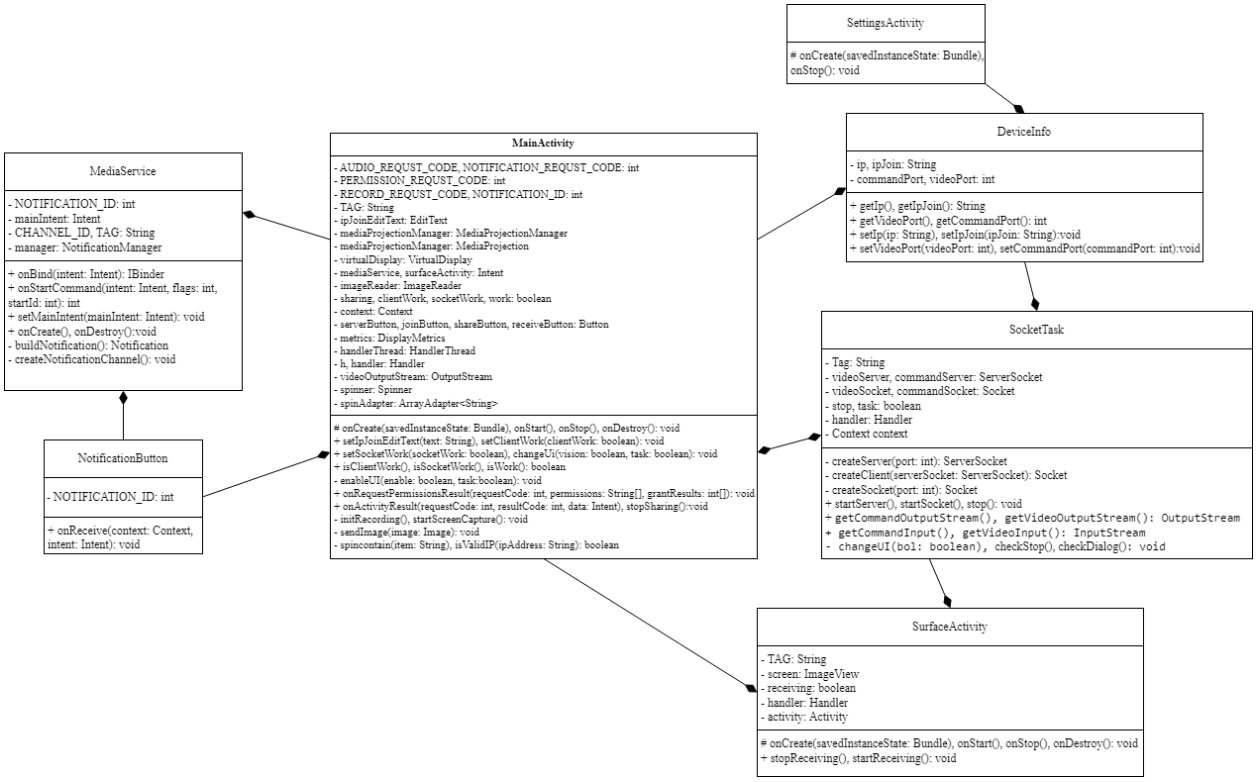


Рис. 2.2 - Діаграма класів *Android* застосунку

«*MainActivity*» вказує *IP*-адреси пристроїв і зберігає їх в «*DeviceInfo*», говорить «*SocketTask*» створити сервер, почати з’єднання, зупинити сервер або з’єднання, говорить «*SurfaceActivity*» почати або зупини прийняття та відображення зображення екрану, говорить «*MediaService*», що при натисканні на повідомлення відкривати головне вікно.

«*SettingsActivity*» вказує порти ліній зв’язку і зберігає їх в «*DeviceInfo*».

«*DeviceInfo*» видає «*SocketTask*» інформацію про лінії зв’язки та *IP*-адреси пристроїв.

«*SocketTask*» видає потоки передачі та прийняття інформації в «*MainActivity*» та «*SurfaceActivity*», говорить «*MainActivity*» змінити інтерфейс та зупинити передачу, а також говорить «*SurfaceActivity*» зупинити прийняття інформації.

«*NotificationButton*» вказує «*MediaService*», що він відповідає за функціонал кнопки, зупиняє трансляцію екрану в «*MainActivity*».

В *Windows* системі «*MainApplication*» являється головним класом, відповідає за початок та кінець роботи системи, створення вікон, захват та передачу зображення екрану.

«*UI*» допоміжний клас відповідає за перехід між вікнами.

Клас «*MainController*» є головним вікном, відповідає за функціонал інтерфейсу що описується файлом *main-viwe.fxml*, за передачу даних об вибраних адресах, початку створення серверу та з'єднання до іншого серверу.

Клас «*ReceivingController*» є вікном, відповідає за функціонал інтерфейсу що описується файлом *receiving-viwe.fxml*, та за прийняття та відображення зображення екрану.

Клас «*SettingsController*» є вікном, відповідає за функціонал інтерфейсу що описується файлом *settings-viwe.fxml*, відповідає за передачу інформації об введених портів ліній зв'язку.

Більш детально взаємозв'язок класів можна побачити в діаграмі класів (рис 2.3)

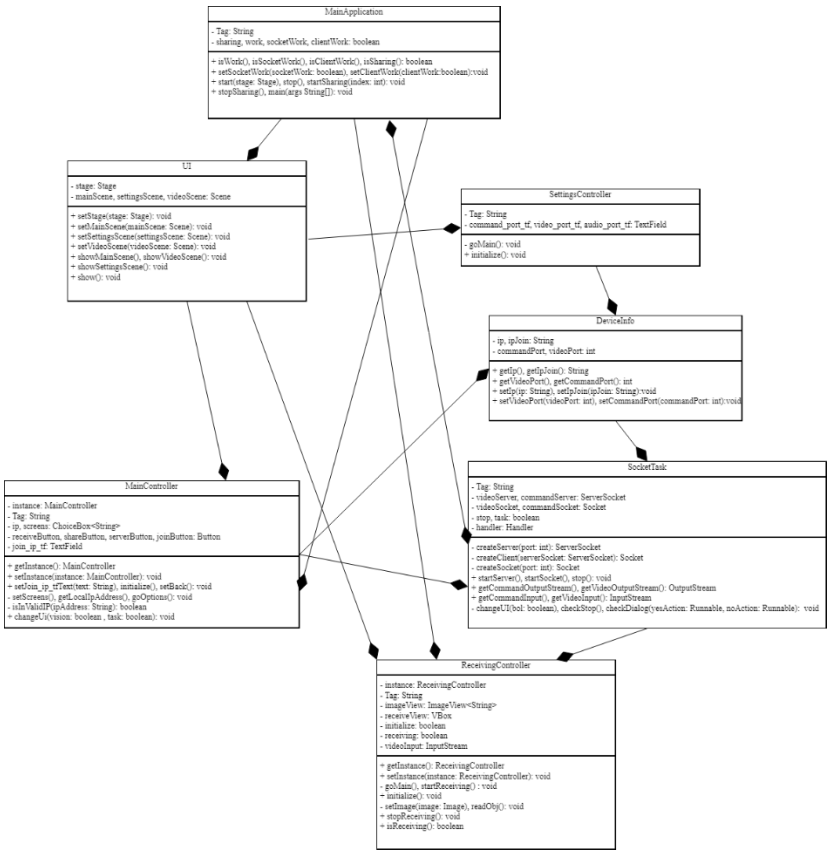


Рис. 2.3 - Діаграма класів *Windows* застосунку

«*MainApplication*» створює вікна та зберігає їх в «*UI*», говорить «*SocketTask*» зупинити сервер або з'єднання, говорить «*ReceivingController*» зупини прийняття зображення екрану та вказує контролер, вказує контролер в «*MainController*»

«*MainController*» вказує *IP*-адреси пристроїв і зберігає їх в «*DeviceInfo*», говорить «*SocketTask*» створити сервер, почати з'єднання, зупинити сервер або з'єднання, говорить «*UI*» відкрити вікна «Налаштування», «Прийняття та відображення зображення екрану».

«*SettingsController*» вказує порти ліній зв'язку і зберігає їх в «*DeviceInfo*», говорить «*UI*» відкрити «Головне вікно».

«*DeviceInfo*» видає «*SocketTask*» інформацію про лінії зв'язки та *IP*-адреси пристроїв.

«*SocketTask*» видає потоки передачі та прийняття інформації в «*MainApplication*» та «*ReceivingController*», говорить «*MainController*» змінити інтерфейс, говорить «*MainApplication*» зупинити передачу зображення екрану, а також говорить «*ReceivingController*» зупинити прийняття інформації.

2.1.3 Діаграма послідовності дій

Діаграма послідовності — різновид діаграми в *UML*. Діаграма послідовності відображає взаємодії об'єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об'єкти та послідовність надісланих повідомлень.

На рисунку 2.4 можна побачити, що для створення серверу користувач спочатку повинен обрати *IP*-адресу для вказання адреси серверу, до якого будуть підключатися пристрої. Після чого натиснути кнопку «Почати», щоб головне вікно сказала механізму підключення створити сервер. Механізм підключення блокує частину інтерфейсу користувача для уникнення появи помилок та створює сервер, який чекає підключення пристрою до нього. При спробі підключитися сервер видає користувачеві запит на дозвіл з'єднатися пристрою з виведеною *IP*-адресу. Якщо користувач підтверджує з'єднання, сервер

підключає лінії зв’язку, дозволяючи отримання даних, та показуючи скриті частини інтерфейсу.

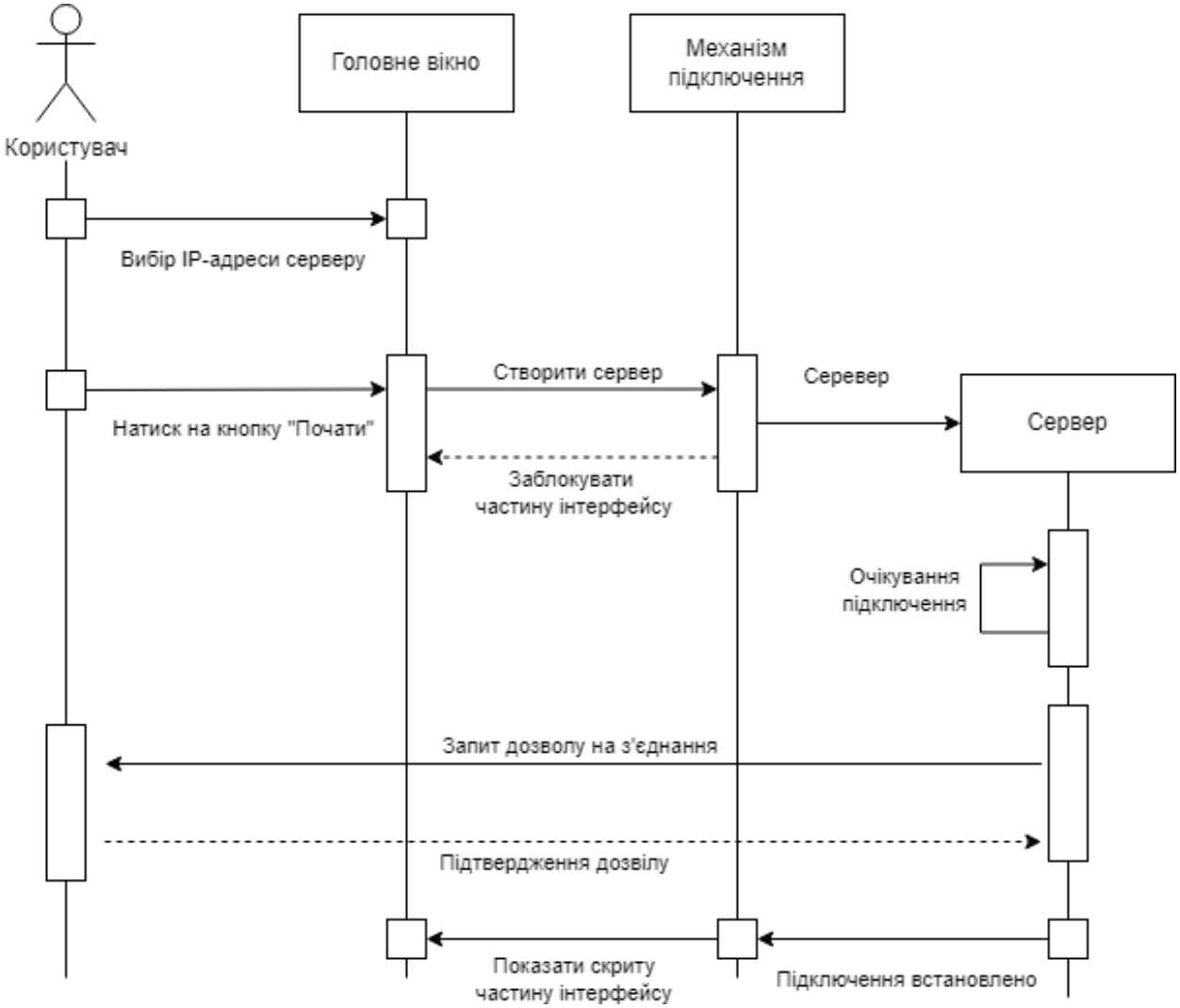


Рис. 2.4 - Діаграма послідовності дій створення серверу

На рисунку 2.5 можна побачити, що для підключення пристрою до сервера користувач повинен ввести *IP*-адресу пристрою з сервером. Після чого натиснути кнопку «Підключитися», щоб головне вікно сказала механізму підключення почати спроби підключення до серверу. Механізм підключення блокує частину інтерфейсу користувача для уникнення появи помилок та з’єднується з сервером підключаючи лінії зв’язку та показуючи скриті частини інтерфейсу для передачі зображення.

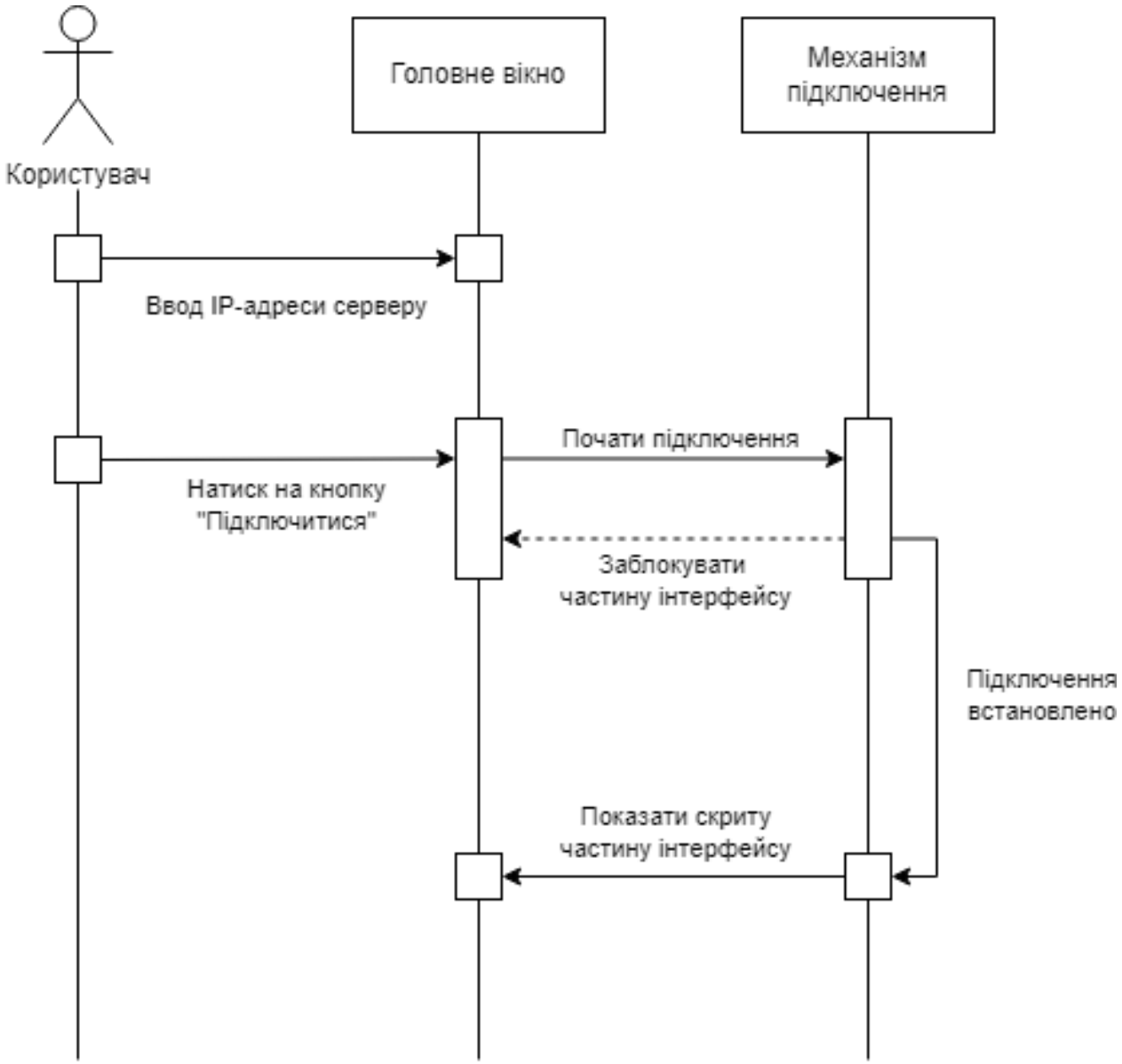


Рис. 2.5 - Діаграма послідовності дій підключення пристрою до серверу

На рисунку 2.6 можна побачити, що для початку прийняття зображення після підключення пристрою до серверу, користувач повинен натиснути кнопку «Прийняти». Головне вікно відкриє користувачу вікно «Прийняття зображення» та скаже системі почати прийняття масиву байтів з серверу, перетворення масиву в зображення та відображення зображення у спеціальному полі в вікні «Прийняття зображення».

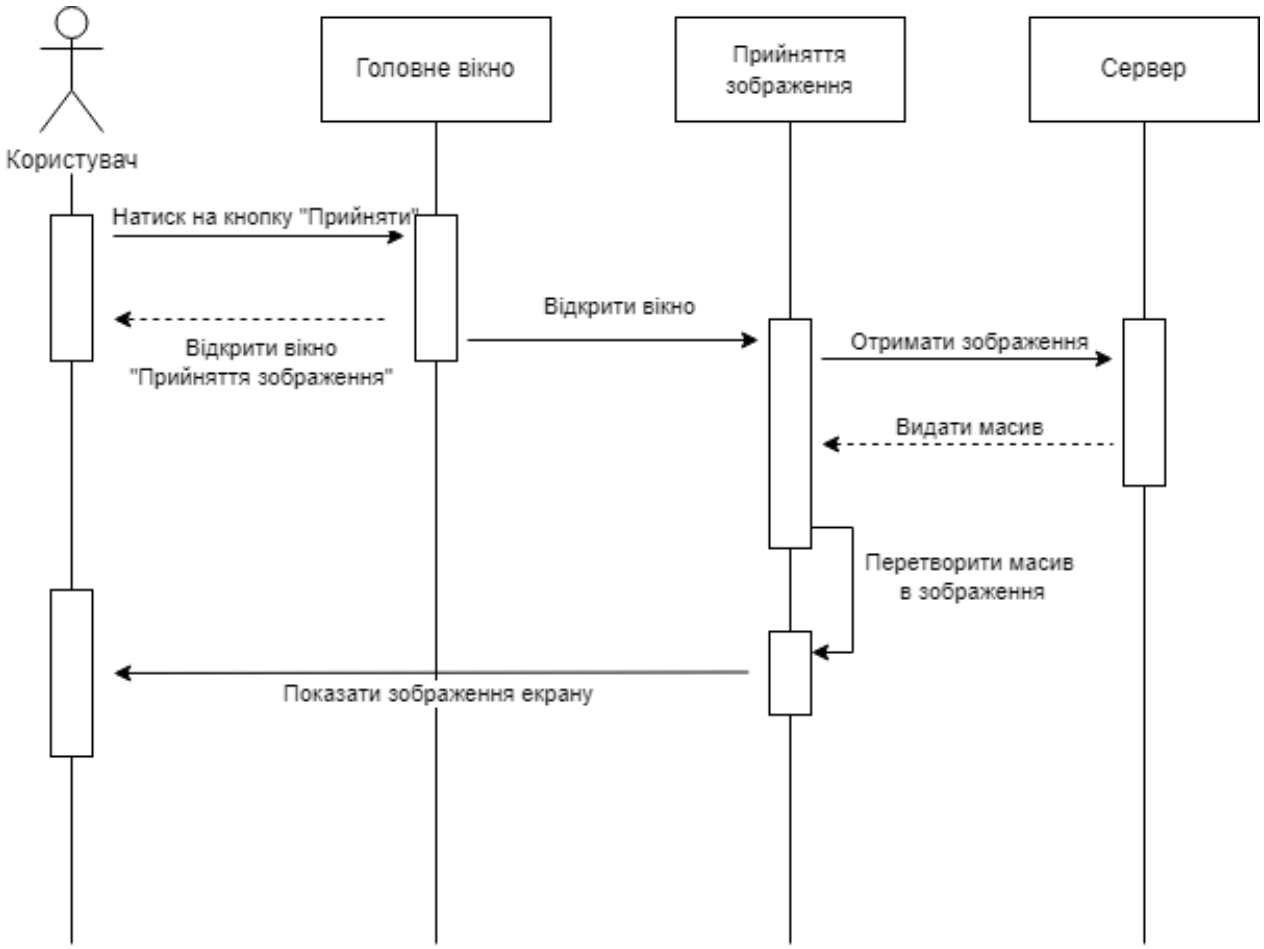


Рис. 2.6 - Діаграма послідовності дій прийняття зображення

На рисунках 2.7 та 2.8 можна побачити, що для початку трансляції в *Windows* застосунку користувач спочатку повинен обрати екран, що буде передаватися. Після натиску на кнопку «Поділитися» в *Android* застосунку виводиться попередження об спробі запису або трансляції зображення екрану, у *Windows* застосунку даного попередження не виводиться і перетворення зображення в масив виконується одразу після натиску кнопки. Якщо користувач підтверджує спробу запису або трансляції зображення екрану, починається захват та перетворення зображення в масив та передача масиву через сервер.

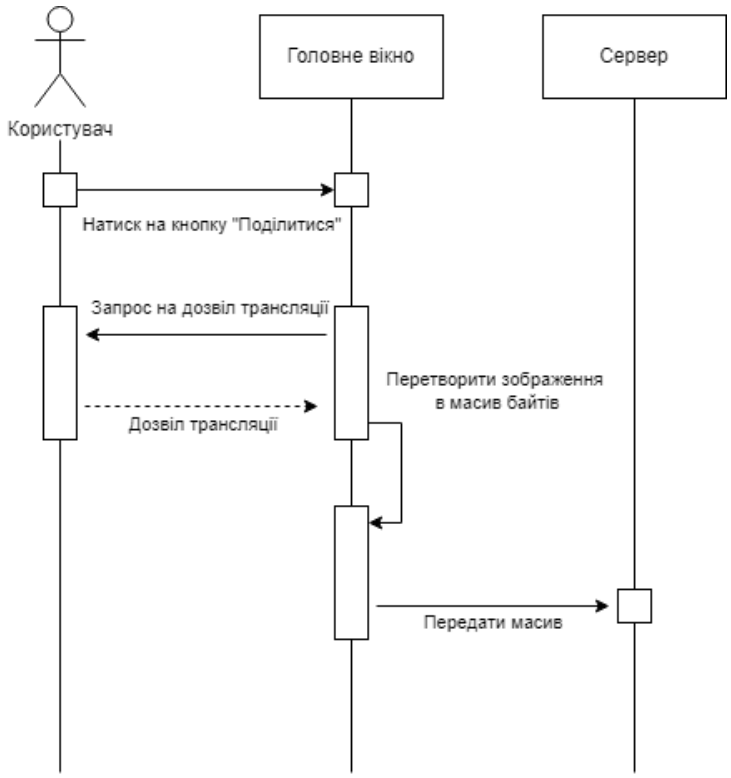


Рис. 2.7 - Діаграма послідовності дій передачі зображення *Android* застосунку

Так як система працює на операційних системах *Android* та *Windows* механізм захвату зображення дещо відрізняється, наприклад до *Windows* пристрою може бути підключено декілька екранів і користувач повинен мати змогу вибирати, який екран він хоче транслювати.

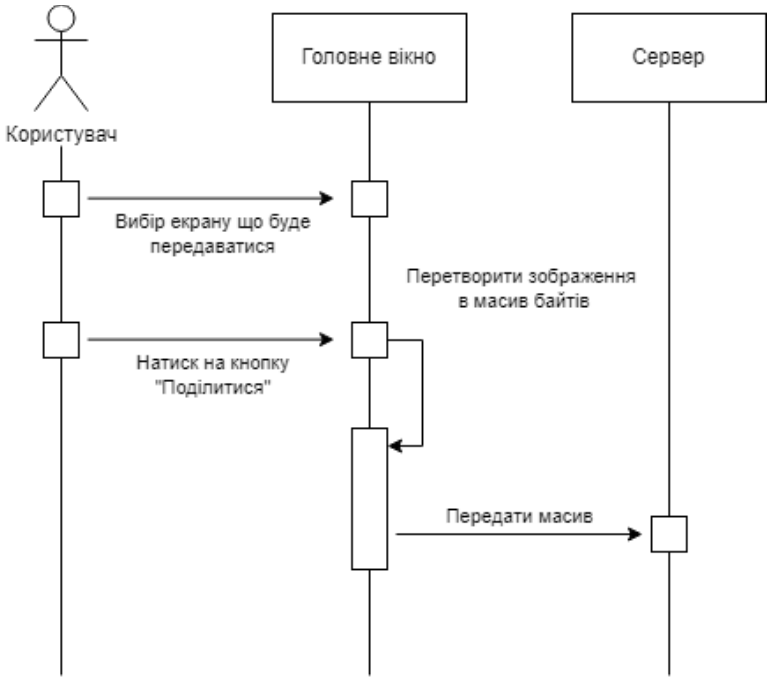


Рис. 2.8 - Діаграма послідовності дій передачі зображення *Windows* застосунку

2.2 Визначення структури системи

Система *Android* застосунку складається з декількох модулів, які включають у себе:

– модуль «Головне вікно» відповідає за запуск та завершення роботи застосунку, за передачу зображення, введення *IP*-адреси пристроїв, за запуск з'єднання пристроїв, відкриття інших вікон застосунку та за зміну користувацького інтерфейсу головного вікна;

– модуль «Налаштування» що відповідає за введення портів зв'язку та за зміну користувацького інтерфейсу вікна налаштувань;

– модуль «Пристрої» що зберігає дані про порти та *IP*-адреси пристроїв;

– модуль «Мережа» що відповідає за з'єднання пристроїв через мережу;

– модуль «Відображення» що відповідає за прийняття та відображення зображення екрану та зміну користувацького інтерфейсу вікна відображення.

Система *Windows* застосунку складається з таких модулів:

– основний модуль відповідає за запуск та завершення роботи застосунку, за передачу зображення, за відображення користувацького інтерфейсу;

– модуль «Головне вікно», відповідає за введення *IP*-адреси пристроїв, за зміну користувацького інтерфейсу головного вікні, за запуск з'єднання пристроїв та відкриття інших вікон застосунку;

– модуль «Налаштування» що відповідає за введення портів зв'язку та зміну користувацького інтерфейс вікна налаштувань;

– модуль «Пристрої» що зберігає дані про порти та *IP*-адреси пристроїв;

– модуль «Мережа» що відповідає за з'єднання пристроїв через мережу;

– модуль «Відображення» що відповідає за прийняття та відображення зображення екрану та зміну користувацького інтерфейсу вікна відображення.

Загалом, моделі роботи усіх модулів застосунків можна відобразити на діаграмах:

Рис. 2.9 – Модель роботи взаємодії модулів *Android* застосунку

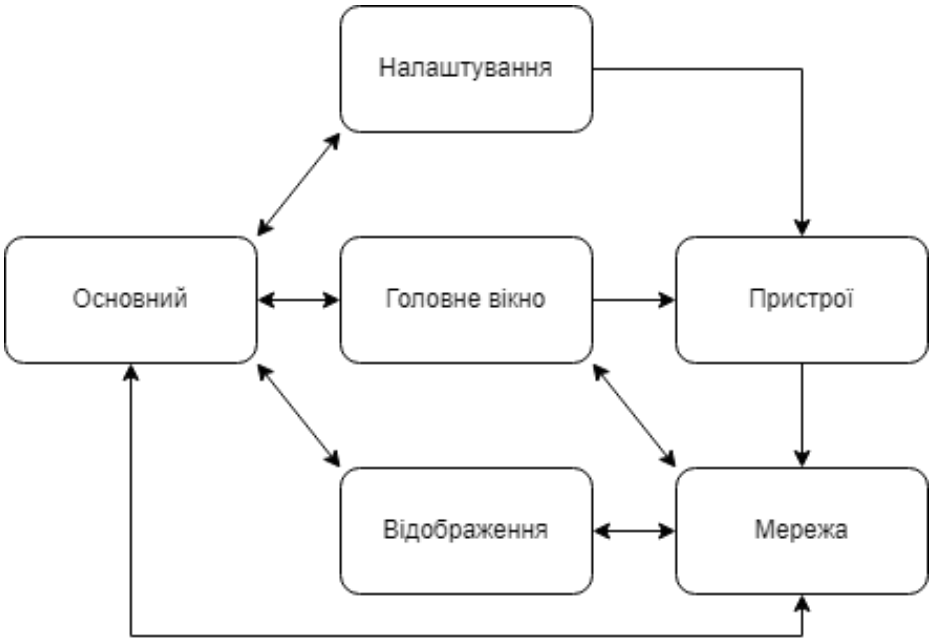


Рис. 2.10 – Модель роботи взаємодії модулів *Windows* застосунку

2.3 Визначення архітектури системи

Для розробки застосунку, потрібно обрати одну з архітектур, за якою буде проводитися його розробка. Основні з них вирізняють:

– нативна архітектура: нативні додатки розробляються спеціально для однієї платформи (*Android* або *iOS*) з використанням мов програмування, які

підтримуються цією платформою (*Java* або *Kotlin* для *Android*, *Swift* або *Objective-C* для *iOS*). Нативні додатки мають високу продуктивність, оптимальну інтеграцію з платформою та можуть повноцінно використовувати всі можливості пристрою;

– гібридна архітектура: гібридні додатки є комбінацією нативних та веб-додатків. Вони розробляються за допомогою *HTML, CSS* та *JavaScript*, а потім "обгортаються" в нативну оболонку, яка дозволяє їм встановлюватися на пристрої та використовувати нативні *API*. Вони зазвичай менш ефективні, ніж нативні додатки, але дозволяють розробникам використовувати один кодовий базіс для різних платформ;

– крос-платформна архітектура: крос-платформні додатки розробляються за допомогою спеціалізованих фреймворків, таких як: *React Native*, *Xamarin*, *Flutter*, *Microsoft Visual Studio* або *IntelliJ IDEA* , які дозволяють використовувати одну кодову базу для створення додатків для різних платформ. Вони зазвичай менш продуктивні, ніж нативні додатки, але можуть забезпечити більш послідовний користувацький досвід на різних пристроях.

Мною було обрано нативну, так як мною використовується *Android Studio* для *Android*-застосунку та *IntelliJ IDEA* для комп’ютерного застосунку.

Також потрібно обрати одну з мережевих архітектур, за якою буде проводитися з’єднання пристроїв та передача данних:

1. Архітектура "клієнт-сервер" передбачає наявність центрального сервера, який надає послуги або ресурси, та клієнтів, які звертаються до сервера для отримання цих послуг або ресурсів. Сервер зазвичай обробляє запити, зберігає дані і виконує обчислення, а клієнти використовують ці дані і обчислювальні результати;

Плюси:

- централізоване управління: Легше контролювати доступ до ресурсів і даних;
- безпека: оскільки всі дані зберігаються на сервері, легше забезпечити їх захист і резервне копіювання;

– простота адміністрування: централізовані оновлення і підтримка спрощують адміністрування;
– масштабованість: легше масштабувати систему за рахунок додавання нових серверів.

Мінуси:

– єдина точка відмови: якщо сервер виходить з ладу, вся система може перестати працювати;
– витрати: потребує витрат на потужний сервер і його обслуговування;
– мережеве навантаження: високе навантаження на сервер і мережу, особливо при великій кількості клієнтів.

2. В одноранговій архітектурі (*Peer-to-Peer, P2P*) кожен вузол (*peer*) мережі є рівноправним і може одночасно виконувати функції клієнта і сервера. Вузли можуть безпосередньо обмінюватися даними без необхідності звертатися до центрального сервера.

Плюси:

– відсутність єдиної точки відмови: вихід з ладу одного вузла не призводить до припинення роботи всієї системи;
– розподілене навантаження: обчислювальні ресурси і мережеве навантаження розподіляються між усіма вузлами;
– масштабованість: система легко масштабується за рахунок додавання нових вузлів;
– низькі витрати: не потребує потужного центрального сервера і витрат на його обслуговування.

Мінуси:

– безпека: складніше забезпечити безпеку даних, оскільки вони розподілені між багатьма вузлами;
– адміністрування: важче адмініструвати і оновлювати систему, оскільки кожен вузол діє незалежно;

– залежність від учасників: продуктивність і надійність системи залежать від якості підключень і продуктивності окремих вузлів.

Порівняння протоколів:

– управління та безпека: архітектура "клієнт-сервер" надає кращий контроль і безпеку даних завдяки централізованому управлінню, тоді як *P2P* важче адмініструвати і забезпечити безпеку.

– надійність: *P2P* більш надійна завдяки відсутності єдиної точки відмови, тоді як "клієнт-сервер" залежить від доступності центрального сервера.

– масштабованість: обидві архітектури можуть бути масштабованими, але *P2P* масштабується природним чином завдяки додаванню нових вузлів, тоді як "клієнт-сервер" потребує додаткових серверів.

– витрати: "клієнт-сервер" потребує значних витрат на потужний сервер і його обслуговування, тоді як *P2P* знижує ці витрати за рахунок використання ресурсів учасників мережі.

Я обрав *P2P* архітектура бо вона є простою у виконані, не потребує потужного центрального сервера і витрат на його обслуговування.

2.4 Визначення базових технологій

Система сама дізнається, які мережі доступні для передачі даних та виводяться їх у випадаючий список у вигляді *IP*-адресів, в якому користувач може обрати один з представлених *IP*-адресів. Порти ліній зв’язку являються числами і вводяться користувачем в окремому вікні застосунку. При помилках та запитах дозволу виводяться діалогові вікна з потрібною інформацією. Для зменшення помилок системи, що спричинені не правильними або випадковими діями користувача, поля вводу даних повинні бути заблоковані.

Система перевіряє правильність введених користувачем текстових або числових даних. Починає виконувати створення серверу та з’єднання пристроїв при натиску на кнопки. Відкриття потрібних вікон при натиску на кнопки. Обробка подій відкриття або закриття вікна, наприклад виконуючи початок та зупинку прийняття зображення екрану. Ввід доступних *IP*-адресів у

випадаючий список. Блокування полів вводу при початку обробки даних даного поля. Зупинка всіх процесів системи при закритті застосунку. Обробка подій при розриві зв’язку або з’єднанні пристроїв, наприклад оповіщення користувача о спробі з’єднання до серверу. Система повинна виводити інформацію про помилки, які з'явилися.

При підключенні пристрою до серверу, на пристрою з сервером виводиться діалогове вікно в якому вказано *IP*-адресу пристрою, що підключається та дві кнопки для підтвердження або відмові у підключенні. Також в *Android* застосунку виводиться вікно для підтвердження початку захвату екрану, що обумовлено потребою операційної системи в даному вікні при спробі захоплення екрану та повідомлення, що застосування веде захоплення.

2.5 Розроблені алгоритми функціональних компонентів

Після натиску користувачем на кнопки «Почати» викликається функція, створення серверу та очікування підключення пристрою до цього серверу, алгоритм якої представлений в додатку А на рисунку А.1.

Функція задає значення глобальним змінним *task, stop, clientWork, agree*. Створюється два віртуальних серверу *commandServer* та *videoServer*, що являються лініями зв’язку. Якщо з’являються помилки в створені серверу *commandServer* та *videoServer* присвоюється значення “null”, через що функція припиняє роботи. Якщо помилок не має і *commandServer* та *videoServer* не дорівнюють “null” одночасно починаються два паралельні цикли.

Перший цикл перевіряє підключення до *commandServer*, якщо до нього не підключено пристрою цикл починається заново, інакше викликається функція *checkDiaolog()*, що показує діалогового вікна «Підтвердження з’єднання», в даній діаграмі показано функцію з *Windows* застосунку і має атрибути, де *yesAction* функція, яка виконується при натиску кнопки «*Yes*» і присвоює значення *agree* “*thrue*”, *noAction* функція, яка виконується при натиску кнопки «*No*», що викликає функцію зупинки з’єднання та видаляє сервер, в *Android* застосунку вона не має атрибутів.

Другий цикл перевіряє значення *agree*, якщо він має значення “*false*”, цикл повторюється, інакше перевіряється підключення до *videoServer*, інакше викликається функція зміни інтерфейсу головного вікна.

Після натиску користувачем на кнопки «Підключитися» викликається функція підключення пристрою до серверу, алгоритм якого показано в додатку А на рисунку А.2.

Функція задає значення глобальним змінним *task*, *stop*, *socketWork*, після чого одночасно починаються два паралельні цикли.

Цикли одночасно намагаються підключитися до двох серверів з різними лініями зв'язку та при підключенні другої лінії викликається функція зміни головного вікна.

2.6 Розроблені шаблони інтерфейсу користувача

На рисунку 2.11 зображено шаблон головного вікна *Android* застосунку з полем вводу, випадаючим списком а трьома кнопками: створити сервер, приєднатися до серверу, відкрити вікно «Налаштувань».

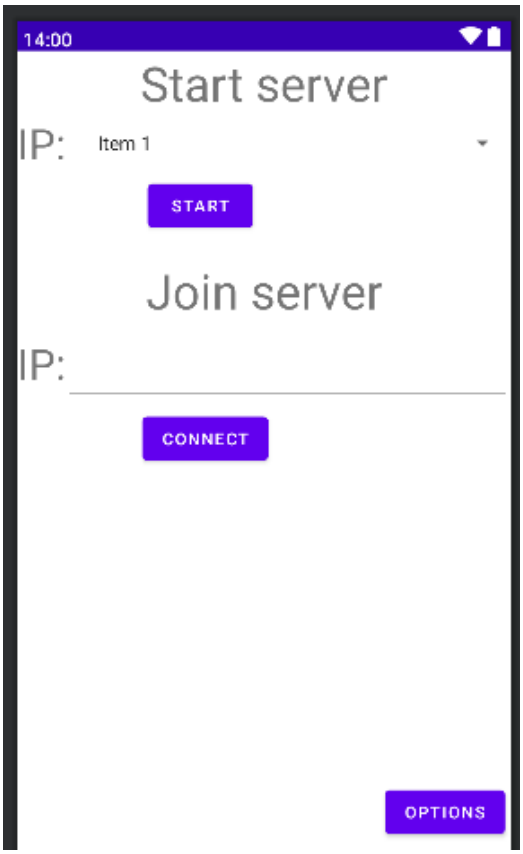


Рис. 2.11 – Шаблон головного вікна *Android* застосунку

На рисунку 2.12 зображено шаблон вікна «Прийняття та відображення зображення екрану» *Android* застосунку з полем виводу зображення а кнопкою для повернення на головне вікно.

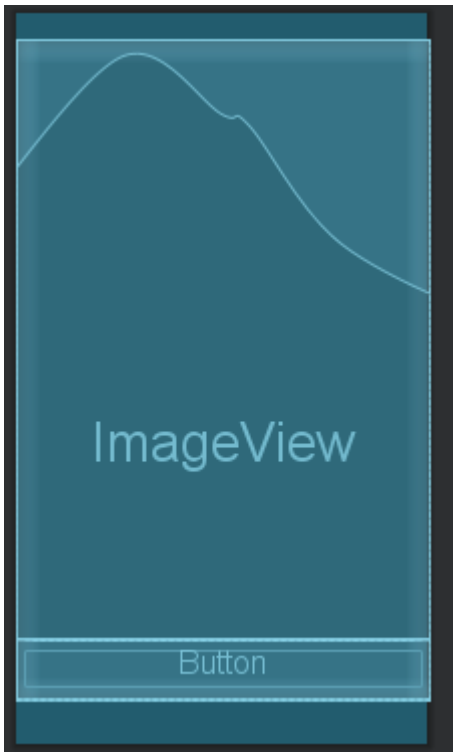


Рис. 2.12 – Шаблон вікна «Прийняття та відображення зображення екрану» *Android* застосунку

На рисунку 2.13 зображено шаблон вікна «Налаштувань» *Android* застосунку з полями вводу портів ліній зв’язку.

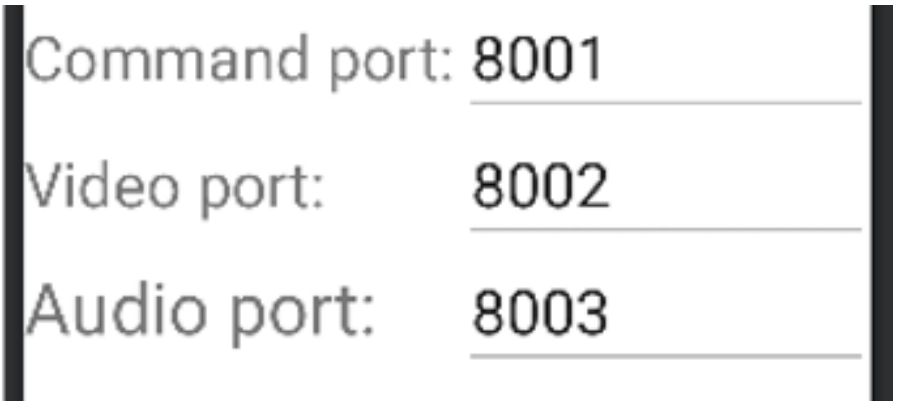


Рис. 2.13 – Шаблон вікна «Налаштувань» *Android* застосунку

Висновок до другого розділу

У процесі проектування було визначено основні та допоміжні функції системи. Представлено діаграми *Use-Case*, діаграми послідовності дій користувача та системи, діаграми класів *Android* та *Windows* застосунків та описані функції та взаємозв'язок кожного класу. Обрано структуру системи, архітектуру розробки та мережеву архітектуру з'єднання та передачі даних. Зроблено декілька шаблонів користувацького інтерфейсу. Описано, як повинна система надавати доступ до даних, які події система повинна обробляти та представлено рішення щодо захисту інформації.

РОЗДІЛ 3
ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Вибір і обґрунтування програмних засобів реалізації

3.1.1 Обрання мови програмування

Від обрання мови програмування залежить середа розробки, розмір коду, час розробки, швидкодія програми, яка частина коду буде перероблено від системи до іншої системи.

Серед багатьох мов програмування, які використовуються у світі задля будь-яких потреб, можна виділити декілька, які найчастіше використовуються для розробки мобільних та комп'ютерних застосунків.

Основні характеристики мови програмування *Java* включають просту мову, яку можна програмувати без тривалої підготовки програмістів, налаштовуючись на поточну практику програмного забезпечення. Основні поняття технології *Java* сприймаються швидко; програмісти можуть бути продуктивними з самого початку.

Мова програмування Java розроблена як об'єктно-орієнтована з самого початку. Програмісти, які використовують мову програмування *Java*, можуть отримати доступ до існуючих бібліотек тестованих об'єктів, які забезпечують функціональність, починаючи від основних типів даних через введення/виведення та мережевих інтерфейсів до наборів інструментів графічного інтерфейсу користувача. Ці бібліотеки можна розширити, щоб забезпечити нову поведінку.[36]

Kotlin — це сучасна, але вже зріла мова програмування, розроблена, щоб зробити розробників щасливішими. Він стислий, безпечний, сумісний із Java та іншими мовами та надає багато способів повторного використання коду між різними платформами для продуктивного програмування. [37]

Технологія *Kotlin Multiplatform* розроблена для спрощення розробки кросплатформних проектів. Це скорочує час, витрачений на написання та

підтримку того самого коду для різних платформ, зберігаючи при цьому гнучкість і переваги рідного програмування. [38]

C++ це мова програмування загального призначення з підтримкою кількох парадигм програмування: об'єктно-орієнтованої, узагальненої, процедурної та ін.

C++ має синтаксис, заснований на синтаксисі *C*. Нововведеннями *C++* порівняно з *C* є:

– підтримка об'єктно-орієнтованого програмування через класи;

– підтримка узагальненого програмування через шаблони;

– доповнення до стандартної бібліотеки;

– додаткові типи даних;

– обробка винятків;

– простори імен;

– вбудовані функції;

– перевантаження операторів;

– перевантаження імен функцій;

– посилання і оператори управління вільно розподіленою пам'яттю. [39]

C# об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи *.NET*.

Синтаксис *C#* близький до *C++* і *Java*. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі *XML*.[40]

Цю мову програмування можна використовувати для розробки *Android*-додатків з використанням платформи *Xamarin*. Розглянемо переваги та недоліки використання *C#* для розробки *Android*-додатків і як *Xamarin* розширює його можливості.

Таблиця 3.1

Порівняння мов програмування

Критерій	*Java*	*Kotlin*	*C++*	*C#*
Парадигми	Об'єктно-орієнтована, процедурна	Об'єктно-орієнтована, функціональна	Об'єктно-орієнтована, процедурна	Об'єктно-орієнтована, процедурна, функціональна
Використання	Веб-розробка, мобільні додатки, серверні програми	Мобільні додатки, серверні програми, веб-розробка	Системне програмування, ігри, високопродуктивні додатки	Веб-розробка, бізнес-додатки, ігри, мобільні додатки
Синтаксис	Подібний до C++	Коротший і більш виразний, ніж у Java	Складний, низькорівневий	Подібний до Java, з елементами спрощення
Управління пам'яттю	Автоматичне (Garbage Collection)	Автоматичне (Garbage Collection)	Ручне	Автоматичне (Garbage Collection)
Основні бібліотеки	Багаті стандартні бібліотеки	Включає всі Java-бібліотеки, розширені новими функціями	Велика кількість бібліотек	Розширена .NET бібліотека

Продовження таблиці 3.1

Критерій	*Java*	*Kotlin*	*C++*	*C#*
Популярність	Висока	Зростаюча	Висока серед системних програмістів	Висока у бізнес-додатках та іграх

Серед усіх цих мов програмування, я обрав саме *Java*. Мій вибір обумовлений тим, що мій досвід у розробці мобільних застосунків є доволі невеликим, але з мовою *Java* я ознайомився під час навчання в університеті та коледжі, тому ця мова мені відома. Основні переваги використання мови *Java* для розробки застосунків, включають у себе:

– платформо незалежність: *Java* працює на різних операційних системах, таких як *Windows*, *macOS* та *Linux*, а також на мобільних пристроях під керуванням *Android*. Це означає, що той самий код можна використовувати на різних платформах без необхідності його переписування;

– велика спільнота розробників: *Java* - одна з найпопулярніших мов програмування, і за його спиною стоїть величезна спільнота розробників. Це означає, що завжди можна знайти підтримку, відповіді на запитання та бібліотеки для вирішення різноманітних завдань;

– багаті бібліотеки та фреймворки: для *Java* існує безліч бібліотек та фреймворків, призначених для розробки програмного забезпечення для мобільних та комп'ютерних пристроїв. Наприклад, для мобільної розробки існує *Android SDK*, який надає широкі можливості для створення різноманітних програм;

– висока продуктивність: *Java* забезпечує високу продуктивність завдяки своїй оптимізованій віртуальній машині (*JVM*) та ефективному складання сміття. Це дозволяє створювати ефективні та чуйні програми;

– безпека: *Java* має механізми безпеки, такі як перевірка типів під час компіляції, обробка винятків та контроль доступу до пам'яті. Це допомагає

запобігти багатьох типових помилок програмування та захистити програму від уразливостей;

– масштабованість: *Java* добре масштабується і підходить як для невеликих проектів, так і для великих програм. Він надає безліч інструментів та практик для управління складністю коду та підтримки його читальності та розширюваності;

– простота навчання та використання: *Java* має відносно простий синтаксис, що спрощує його вивчення для новачків та підвищує продуктивність розробників. Крім того, багато IDE (середовища розробки) надають широкі можливості автодоповнення коду та налагодження, що спрощує процес розробки;

– багато поточність: *Java* підтримує багато поточність із коробки, що дозволяє створювати багатозадачні програми, що ефективно використовують ресурси пристрою та забезпечують чуйний інтерфейс.

Ці переваги роблять *Java* привабливим вибором розробки програмного забезпечення як мобільних, так комп'ютерних пристроїв.

3.1.2 Обрання середовища розробки

Серед *IDE* для андроїд застосунків найбільшою популярністю користуються *Android Studio* та *Microsoft Visual Studio*, для комп’ютерних застосунків найбільшою популярністю користуються *IntelliJ IDEA* та *Microsoft Visual Studio*.

Android Studio– заснована на програмному забезпеченні *IntelliJ IDEA* від *JetBrains* і була випущена як заміна *Eclipse* як офіційна *IDE* для розробки додатків *Android*.

Це середовище пропонує повний інструмент для розробки та налагодження програм для мобільної операційної системи *Google*. За допомогою нього можна виконувати редагування коду, налагодження, використовувати інструменти продуктивності, воно має гнучку систему компіляції та миттєве створення та розгортання, що дозволяє зосередитись на створенні додатків.

Особливості *Android Studio*:

1. Розширений редактор макетів: *WYSIWYG*, здатність працювати з *UI* компонентами за допомогою *Drag-and-Drop*, функція попереднього перегляду макета на декількох конфігураціях екрана.

2. Складання додатків, засноване на *Gradle*.

3. Різні види збірок та генерація декількох *.apk* файлів

4. Рефакторинг коду

5. Статичний аналізатор коду (*Lint*), що дозволяє знаходити проблеми продуктивності, несумісності версій та інше.

6. Шаблони основних макетів та компонентів *Android*.

7. Підтримка розробки додатків для *Android Wear* та *Android TV*.

8. Вбудована підтримка *Google Cloud Platform*, яка включає інтеграцію з сервісами *Google Cloud Messaging* і *App Engine*.

9. В *Android Studio* 3.0 за стандартом включені інструменти мови *Kotlin*, засновані на *JetBrains IDE* [41].

IntelliJ IDEA— комерційне інтегроване середовище розробки для різних мов програмування (*Java, Python, Scala, PHP* та ін.) від компанії *JetBrains*.

IntelliJ IDEA надає розробникам широкі можливості розробки засобів для ефективної та спільної розробки.

IntelliJ IDEA має:

1. Потужний редактор коду: *IntelliJ IDEA* має ряд потужних інструментів для редагування коду, таких як автоматичне доповнення коду, вбудовані підказки, перейменування символів та інші.

2. Широкі можливості рефакторингу: інтегровані інструменти рефакторингу дозволяють легко змінювати структуру вашого коду без ризику введення помилок. Це включає в себе перейменування, виділення в метод, витягування константи, оптимізацію імпорту, вилучення методу та багато іншого.

3. Підтримка мов програмування: *IntelliJ IDEA* підтримує не лише *Java*, але також інші мови програмування, такі як *Kotlin, Scala, Groovy, JavaScript*,

TypeScript, SQL тощо. Це робить його відмінним вибором для проєктів, які використовують різні технології.

4. Інтеграція з іншими інструментами: *IntelliJ IDEA* має широку підтримку інших інструментів розробки, таких як системи контролю версій (*Git, SVN*), системи збірки (*Maven, Gradle*), середовища виконання (*Tomcat, JBoss*) та інші.

5. Багатофункціональність: *IntelliJ IDEA* має багато корисних функцій, таких як вбудований дебагер, інтегрований збірник та інші інструменти для вивчення коду.

6. Підтримка плагінів: *IntelliJ IDEA* дозволяє розширювати свою функціональність за допомогою різних плагінів. Це дозволяє налаштувати середовище розробки під свої потреби та вимоги.

Серед розглянутих середовищ найкращим вибором буде *Android Studio* та *IntelliJ IDEA* через свою підтримку мови програмування *Java*, підтримкою багатьох операційних систем та значно більш зручним інтерфейсом. Також до переваг перед *Android Studio*, можна віднести вбудовану підтримку з сервісами *Google*. Крім цього в *Android Studio* та *IntelliJ IDEA* дозволяє підключати додаткові бібліотеки, як створені іншими користувачами платформи так і офіційні. Можна виділити набагато більшу базу різних посібників, які дозволяють розробникам швидко знайти шляхи реалізації поставлених задач.

3.2 Опис класів *Android* застосунку

Застосунок складається з головного вікна, вікна «Налаштувань» та вікна «Прийняття та відображення зображення екрану». У головному вікні користувач вибирає та вводить *IP*-адреси, починає створення та з’єднання до серверу, починає захват та передачу зображення екрану, перехід між іншими вікнами. У вікні «Налаштувань» користувач вводить порти ліній зв’язку. У вікні «Прийняття та відображення зображення екрану» відображається зображення екрану іншого пристрою у окремому полі.

3.2.1 *MainActivity*

«*MainActivity*» являється головним класом та вікном, відповідає за передачу даних об вибраних адресах, початку з'єднання з пристроєм, перехід між вікнами, захват та передачу зображення екрана.

int RECORD_REQUEST_CODE змінна, є кодом або посиланням дозволу для захвату та передачі зображення екрану.

int PERMISSION_REQUEST_CODE змінна, є кодом або посиланням дозволу передньо планових сервісів.

int NOTIFICATION_REQUEST_CODE змінна, є кодом або посиланням дозволу на вивід повідомлень.

int NOTIFICATION_ID змінна, є кодом або посиланням на повідомлення про здійснення захвату та передачі зображення екрану.

String TAG змінна, використовується для вказання місця відлагодження в журналі відлагодження.

EditText ipJoinEditText змінна-об'єкт поля для вводу *IP*-адресу пристрою з сервером.

MediaProjectionManager mediaProjectionManager змінна, що керує отриманням певних типів токенів *MediaProjection* та дозволяє захват екрану.

MediaProjection mediaProjection змінна, є маркером, що надає додаткам можливість знімати вміст екрана.

VirtualDisplay virtualDisplay змінна, є віртуальним екраном захопленого *mediaProjection*.

Intent mediaService змінна, є абстрактний опис операції, передньо планового сервісу, що створює та показає повідомлення про здійснення захвату та передачі зображення екрану.

ImageReader imageReader змінна, використовується для отримання зображення з *virtualDisplay* захопленого екрану.

boolean sharing змінна-прапор відбування захоплення та передачі зображення екрану.

boolean clientWork змінна-прапор створення серверу та підключення пристрою до серверу.

boolean socketWork змінна-прапор підключення даного пристрою до серверу.

boolean work змінна-прапор для відстеження чи працює застосунок.

DisplayMetrics metrics змінна, використовується для відстежування пареметрів екрану.

Intent surfaceActivity змінна, абстрактний опис операції, використовується для відкриття вікна «Прийняття та відображення зображення екрану»

HandlerThread handlerThread змінна, фоновий потік передачі зображення.

Handler h змінна, використовується для виконання дій в фоновому потоці *handlerThread*.

Handler handler змінна, використовується для виконання дій з вторинного потоку в головному потоці, наприклад змінна інтерфейсу.

OutputStream videoOutputStream змінна, є вихідним потоком даних, використовується для передачі зображення екрану.

Spinner spinner змінна, є об’єктом користувацького інтерфейсу з випадаючим списком з *IP*-адресами даного пристрою.

ArrayAdapter<String> spinAdapter змінна, допоміжний об’єкт *spinner*, зберігає список із *IP*-адресами даного пристрою, який відображає *spinner*.

void onCreate(Bundle savedInstanceState) метод, що викликається, коли починається діяльність вікна, параметр *savedInstanceState* використовується, якщо активність повторно ініціалізується після попереднього вимкнення і містить дані, які він нещодавно надав у *onSaveInstanceState(Bundle)*.

void setIpJoinEditText(String text) метод, що вводить текст з праметру *text* у поле *ipJoinEditText*

boolean isClientWork() метод, що передає значення *clientWork*.

void setClientWork(boolean clientWork) метод, що задає значення глобальній змінній *clientWork* значення локальної *clientWork*.

boolean isClientWork() метод, що передає значення *clientWork.*

void setSocketWork (boolean socketWork) метод, що задає значення глобальній змінній *socketWork* значення локальної *socketWork.*

boolean isWork() метод, що передає значення *work.*

void changeUi(boolean vision, boolean task) метод, що змінює інтерфейс головного вікна, де *vision* вказує чи буде видно якісь елементи інтерфейсу, *task* вказує яка операція проводиться, створення серверу та підключення до нього або тільки підключення до серверу.

void enableUI(boolean enable, boolean task) метод, що змінює інтерфейс головного вікна, де *enable* вказує чи зможе користувач взаємодіяти з якимось елементами інтерфейсу, *task* вказує яка операція проводиться, створення серверу та підключення до нього або тільки підключення до серверу.

void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) метод, використовується для вказання дозволу на передньо планові сервіси та на вивід повідомлень. Де *requestCode* це код запиту, *permissions* це запитані дозволи, *grantResults* це результати надання для відповідних дозволів.

void startScreenCapture() метод, що починає захват екрану.

void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) метод, що виводить попередженням об спробі запису або трансляції зображення екрану. Де *requestCode* це код запиту, *resultCode* це результат надання дозволу, *data* це намір, який може повертати дані результату куористувача.

void initRecording() метод, що починає передачу зображення.

void sendImage(Image image) метод, що передає об’єкт image через *videoOutputStream.*

void stopSharing() метод, що зупиняє захват та передачу зображення.

void onStart() метод, що викликається автоматично при відкритті вікна, використовується для початку заповнення списку *IP*-адресів даного пристрою.

void onStop() метод, що викликається коли згортається вікно.

void onDestroy() метод, що викликається коли закривається вікно або застосунок.

boolean isValidIP(String ipAddress) метод, що перевіряє чи є правильного виду *IP*-адреса в *ipAddress*.

3.2.2 *SocketTask*

«*SocketTask*» цей клас відповідає за з'єднання пристроїв між собою та за доступ до ліній передачі даних.

String TAG змінна, використовується для вказання місця відлагодження в журналі відлагодження.

ServerSocket commandServer змінна, є об'єктом командного серверу.

ServerSocket videoServer змінна, є об'єктом серверу, що передає зображення.

boolean stop змінна-прапор який говорить чи було прийнято команду зупинки від іншого пристрою.

boolean task змінна-прапор, що вказує яка операція проводиться, створення серверу та підключення до нього або тільки підключення до серверу.

boolean agree змінна-прапор, що вказує, чи дозволив користувач підключення даного пристрою до серверу.

Context context змінна, що є інтерфейсом до глобальної інформації про середовище застосунку, використовується для створення та відображення діалогового вікна підтвердження підключення пристрою до серверу.

Handler handler змінна, використовується для виконання дій в головному потоці.

ServerSocket createServer(int port) метод, використовується для створення серверу з лінією зв'язку *port*.

Socket createClient(ServerSocket serverSocket) метод, що підключає пристрій до *serverSocket*.

Socket createSocket(int port) метод, що підключає пристрій до серверу з лінією зв'язку *port*.

void startServer() метод, що створює сервер та підключає пристрій до цього серверу.

void checkDialog() метод, що створює та відображає діалогового вікно підтвердження підключення.

void startSocket() метод, що підключає пристрій до серверу.

void checkStop() метод, що перевіряє чи прийшла команда зупинки від іншого пристрою.

void stop() метод, що зупиняє з'єднання пристрою та серверу.

OutputStream getCommandOutputStream() метод, що видає вхідний потік командних даних.

OutputStream getVideoOutputStream() метод, що видає вхідний потік даних, що використовується для передачі зображення екрану.

InputStream getCommandInput() метод, що видає вихідний потік командних даних.

InputStream getVideoInput() метод, що видає вихідний потік даних, що використовується для отримання зображення екрану.

void changeUi(boolean bol) метод, що змінює інтерфейс головного вікна, де *bol* вказує стан видимості елементів інтерфейсу.

void setContext(Context context) метод, що задає значення глобальній змінній *context* значення локальної *context*.

3.2.3 *SurfaceActivity*

«*SurfaceActivity»* відповідає за прийняття та відображення зображення екрану.

String TAG змінна, використовується для вказання місця відлагодження в журналі відлагодження.

ImageView screen змінна, що є полем користувацького інтерфейсу, відображає прийняте зображення.

boolean receiving змінна-прапор, що ведеться прийняття та відображення екрану.

Handler handler змінна, використовується для виконання дій в головному потоці.

Activity activity змінна, є вікном «Передачі та відображення зображення екрану»

void onCreate(Bundle savedInstanceState) метод, що викликається, коли починається діяльність вікна, *savedInstanceState* використовується, якщо активність повторно ініціалізується після попереднього вимкнення і містить дані, які він нещодавно надав у *onSaveInstanceState(Bundle).*

void onStart() метод, що викликається автоматично при відкритті вікна.

void startReceiving() метод, що використовується для прийняття і відображання зображення екрану.

void stopReceiving() метод, що зупиняє прийняття і відображання зображення екрану.

void onStop() метод, що викликається коли згортається вікно.

void onDestroy() метод, що викликається коли закривається вікно.

3.3 Опис класів *Windows* застосунку

Застосунок складається з головного вікна, вікна «Налаштувань» та вікна «Прийняття та відображення зображення екрану».

У головному вікні користувач вибирає та вводить *IP*-адреси, починає створення та з'єднання до серверу, починає захват та передачу зображення екрану, перехід між іншими вікнами.

У вікні «Налаштувань» користувач вводить порти ліній зв'язку.

У вікні «Прийняття та відображення зображення екрану» відображається зображення екрану іншого пристрою у окремому полі.

3.3.1 *MainApplication*

«*MainApplication*» являється головним класом, відповідає за початок та кінець роботи системи, створення вікон, захват та передачу зображення екрану.

String Tag змінна, використовується для вказання місця відлагодження в журналі відлагодження.

boolean sharing змінна-прапор відбування захоплення та передачі зображення екрану.

boolean clientWork змінна-прапор створення серверу та підключення пристрою до серверу.

boolean socketWork змінна-прапор підключення даного пристрою до серверу.

boolean work змінна-прапор для відстеження чи працює застосунок.

boolean isClientWork() метод, що передає значення *clientWork.*

void setClientWork(boolean clientWork) метод, що задає значення глобальній змінній *clientWork* значення локальної *clientWork.*

boolean isClientWork() метод, що передає значення *clientWork.*

void setSocketWork (boolean socketWork) метод, що задає значення глобальній змінній *socketWork* значення локальної *socketWork.*

boolean isWork() метод, що передає значення *work.*

boolean isSharing() метод, що передає значення *sharing*.

void start(Stage stage) метод, що викликається при виконані команди *launch(),* де *stage* є платформою на якій показуються вікна.

void stop() метод, що викликається перед закриттям застосунку, зупиняє всі процеси в застосунку.

void startSharing(int index) метод, що захоплює та передає зображення екрану з індексом *index.*

void stopSharing() метод, що зупиняє захват та передачу зображення.

void main(String[] args) метод, що викликається при відкритті застосунку, де *args* допоміжні аргументи запуску застосунку.

3.3.2 *MainController*

«*MainController*» є головним вікном, відповідає за передачу даних об вибраних адресах, початку створення серверу та з'єднання до іншого серверу.

MainController instance змінна, є екземпляр даного класу.

ChoiceBox<String> ip змінна, є об'єктом користувацького інтерфейсу є випадаючим списком з *IP*-адресами даного пристрою.

ChoiceBox<String> screens змінна, є об'єктом користувацького інтерфейсу є випадаючим списком з індексами екранів даного пристрою.

Button receiveButton змінна, є об'єктом користувацького інтерфейсу є кнопкою, що відкриває вікно «Прийняття та відображення зображення екрану».

Button shareButton змінна, є об'єктом користувацького інтерфейсу є кнопкою, що починає захват та передачу зображення екрану.

Button serverButton змінна, є об'єктом користувацького інтерфейсу є кнопкою, що створює сервер та підключає пристрій до цього серверу.

Button joinButton змінна, є об'єктом користувацького інтерфейсу є кнопкою, що підключає пристрій до серверу.

String Tag змінна, використовується для вказання місця відлагодження в журналі відлагодження.

TextField join_ip_tf змінна, є об'єктом поля для вводу *IP*-адресу пристрою з сервером.

MainController getInstance() метод, що передає значення *instance*.

void setInstance(MainController instance) метод, що задає значення глобальній змінній *instance* значення локальної *instance*.

void goOptions() метод, що відкриває вікно «Налаштувань».

void initialize() метод, що викликається при створені вікна, задає функціонал кнопок, починає заповнювати випадаючі списки.

void setBack() метод, що повертає інтерфейс головного вікна в початковий варіант.

void setIpJoinEditText(String text) метод, що вводить текст *text* у поле *join_ip_tf*.

void setScreens() метод, що заповнює випадаючий список *screens*.

void getLocalIpAddress() метод, що заповнює випадаючий список *ip*.

boolean isValidIP(String ipAddress) метод, що перевіряє чи є правильного виду *IP*-адреса в *ipAddress*.

void changeUi(boolean vision, boolean task) метод, що змінює інтерфейс головного вікна, де *vision* вказує чи буде видно якісь елементи інтерфейсу, *task* вказує яка операція проводиться, створення серверу та підключення до нього або тільки підключення до серверу.

3.3.3 *ReceivingController*

«*ReceivingController*» є вікном, відповідає за прийняття та відображення зображення екрану.

String Tag змінна, використовується для вказання місця відлагодження в журналі відлагодження.

ReceivingController instance змінна, є екземпляром даного класу.

boolean initialize змінна-прапор, чи було ініціалізоване вікно.

ImageView imageView змінна, є об'єктом користувацького інтерфейсу що відображає прийняте зображення.

VBox receiveView змінна, є вертикальним контейнером вікна «Прийняття та відображення зображення екрану», потрібен для зчитування зміни розмірів вікна і задня нового розміру *imageView*.

boolean receiving змінна-прапор, що ведеться прийняття та відображення екрану.

InputStream videoInput змінна, є вхідним потоком даних, що використовується для отримання зображення екрану.

void setImage(Image image) метод, що відображає *image* в *imageView*.

ReceivingController getInstance() метод, що передає значення *instance*.

void setInstance(ReceivingController instance) метод, що задає значення глобальній змінній *instance* значення локальної *instance*.

void goMain() метод, що відкриває головне вікно.

void initialize() метод, що викликається при створені вікна, налаштовую *receiveView* для зміни розмірів *imageView* разом з розміром вікна.

void startReceiving() метод, що починає прийняття та відображення зображення екрану.

void readObj() метод, що зчитує об’єкт зображення з *videoInput*.

void stopReceiving() метод, що зупиняє прийняття та відображення зображення екрану.

boolean isReceiving() метод, що передає значення *receiving*.

3.3.4 *SocketTask*

«*SocketTask*», цей клас відповідає за з’єднання пристроїв між собою та за доступ до ліній передачі даних.

String Tag змінна, використовується для вказання місця відлагодження в журналі відлагодження.

ServerSocket commandServer змінна, є об’єктом командного серверу.

ServerSocket videoServer змінна, є об’єктом серверу, що передає зображення.

boolean stop змінна-прапор, який говорить чи було прийнято команду зупинки від іншого пристрою.

boolean task змінна-прапор, що вказує яка операція проводиться, створення серверу та підключення до нього або тільки підключення до серверу.

boolean agree змінна-прапор, що вказує, чи дозволив користувач підключення даного пристрою до серверу.

Handler handler змінна, використовується для виконання дій в головному потоці.

ServerSocket createServer(int port) метод, використовується для створення серверу з лінією зв’язку *port*.

Socket createClient(ServerSocket serverSocket) метод, що підключає пристрій до *serverSocket*.

Socket createSocket(int port) метод, що підключає пристрій до серверу з лінією зв’язку *port*.

void startServer() метод, що створює сервер та підключає пристрій до цього серверу.

void checkDialog(Runnable yesAction, Runnable noAction) метод, що створює та відображає діалогового вікно підтвердження підключення, де *yesAction* функція, яка виконується при натиску кнопки «*Yes*», *noAction* функція, яка виконується при натиску кнопки «*No*».

void startSocket() метод, що підключає пристрій до серверу.

void checkStop() метод, що перевіряє чи прийшла команда зупинки від іншого пристрою.

void stop() метод, що зупиняє з'єднання пристрою та серверу.

OutputStream getCommandOutputStream() метод, що видає вихідний потік командних даних.

OutputStream getVideoOutputStream() метод, що видає вихідний потік даних, що використовується для передачі зображення екрану.

InputStream getCommandInput() метод, що видає вхідний потік командних даних.

InputStream getVideoInput() метод, що видає вхідний потік даних, що використовується для отримання зображення екрану.

void changeUi(boolean bol) метод, що змінює інтерфейс головного вікна, де *bol* вказує стан видимості елементів інтерфейсу.

3.4 Тестування та відлагодження розробленого забезпечення

Для тестування та відлагодженя *Windows* застосунку в *IntelliJ IDEA* я використовую команду *System.out.println()*, дана команда виводить текст в дужках в командну строку.

```
MainApplication main()
MainApplication start
MainController initialize
SettingsController initialize
MainController serverButton event
SocketTask startServer
SocketTask startServer commandSocket thread start
SocketTask startServer videoSocket thread start
MainApplication stop()
SocketTask stop
SocketTask t end
SocketTask startServer commandSocket thread end
ReceivingController stopReceiving
MainApplication stopSharing
MainController changeUI vision false task true
MainApplication main() -> launch()
```

Рис. 3.1 – Відлагодження в командній строчці

Для тестування та відлагодженя *Android* застосунку використовуються *Log* або журнал відлагоженя. В журналі можна виділяти текст червоним, робити фільтр в записах журналу. В журнал запис додається командою *Log.d()*, а для виділеного тексту *Log.e()*.

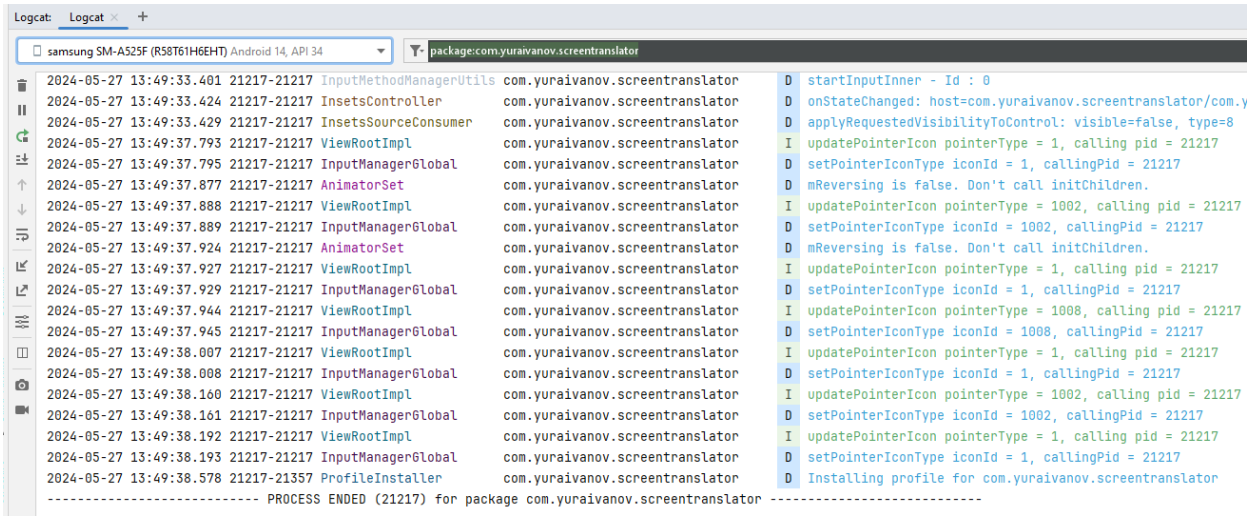


Рис. 3.2 – Журнал відлагодженя *Android* застосунку

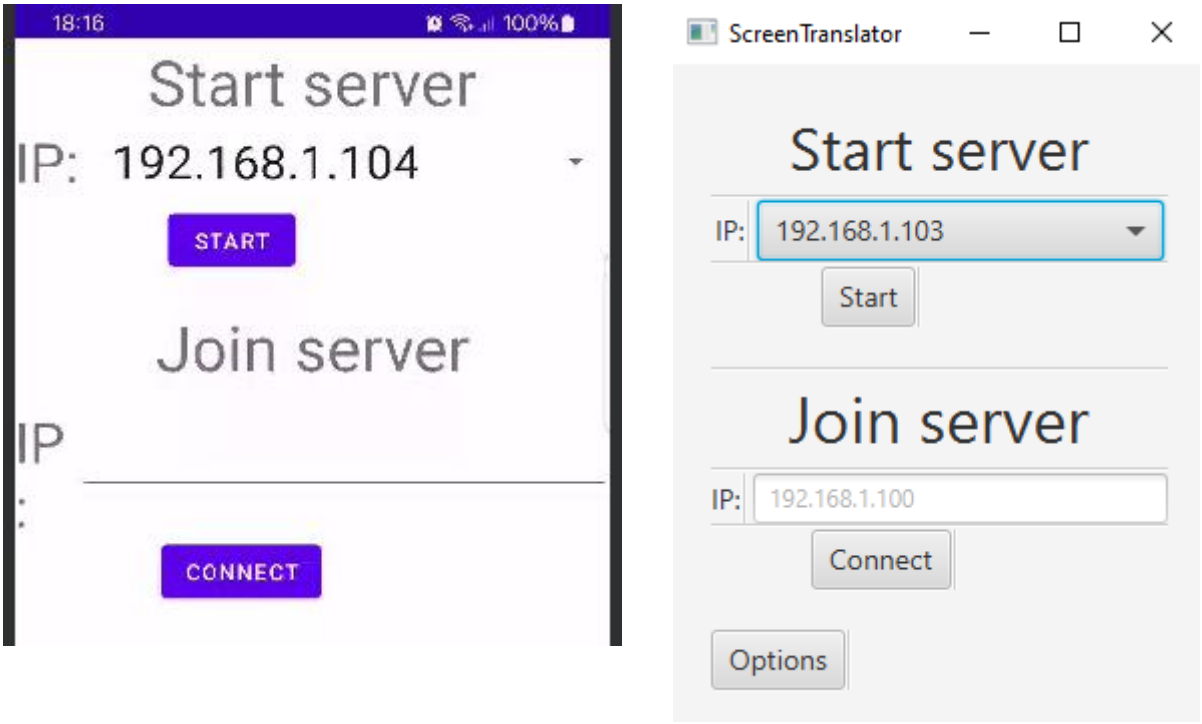
3.5 Збірка та розгортання функціональних одиниць

Для роботи застосунку потрібно для *Android* пристрою скачати встановлючий файл з розширенням *apc* та запустити його на *Android* пристрою, щоб

встановити программу, а для *Windows* пристрою встановити *exe* на будь-який носій даних, це і є застосунок. Після встановлення потрібно підключити пристрої до однієї мережі, це можна зробити як через маршрутизатор, так і підключивши *Windows* або інший *Android* пристрій до точці доступу *Android* пристрою.

Інтерфейс та кроки використання застосунку майже однакові як для *Android* так і для *Windows* пристроїв, різниця буде вказана пізніше.

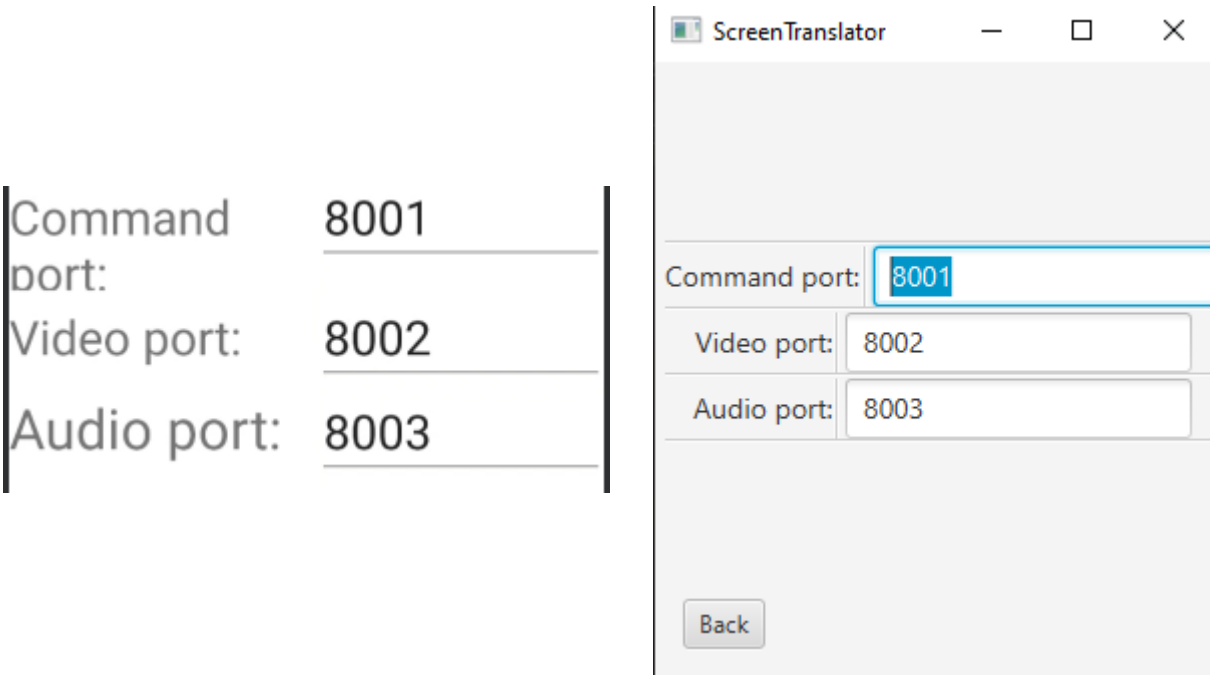
При запуску застосунку, користувач одразу потрапляє у головне вікно (рис. 3.3) де користувач може перейти до вікна «Налаштувань» (рис. 3.4) натиснувши кнопку «*Options*», де користувач може ввести потрібні порти ліній зв’язку, а потім повернутися на головне вікно.



а) б)

Рис. 3.3 – Головне вікно

а) *Android* застосунку б) *Windows* застосунку

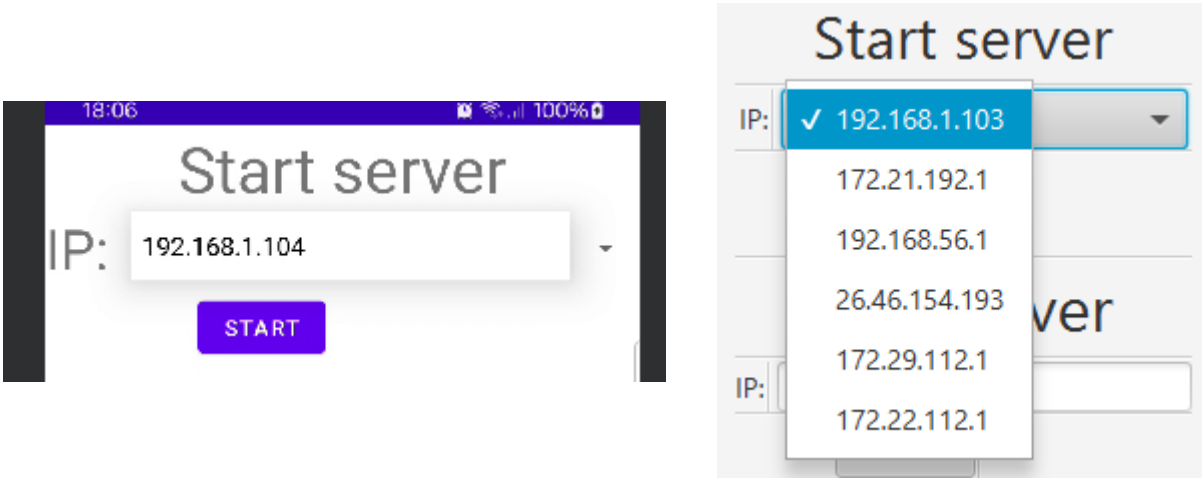


а) б)

Рис. 3.4 – Вікно «Налаштування»

а) *Android* застосунку б) *Windows* застосунку

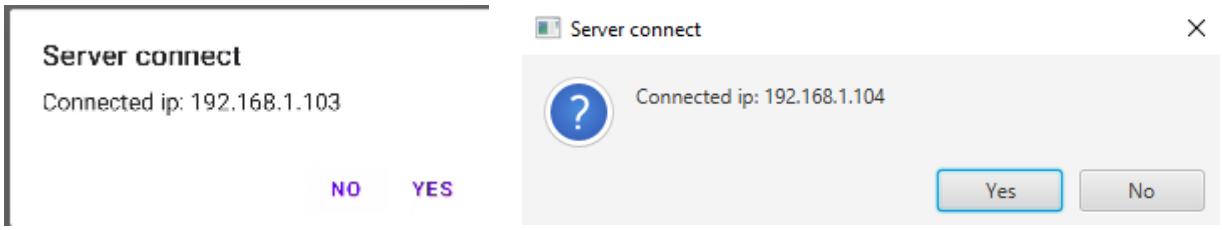
Щоб інші пристрої могли з’єднатися з цим простоєм користувач повинен обрати *IP*-адресу даного пристрою у випадаючому списку (рис 3.5), в полі під написом «*Start server*» (почати сервер), після натискає кнопку «*Start*» (почати) і почнеться процес з’єднання, кнопка змінне свою назву на «*Stop*» (зупинити). При спробі підключення до серверу з’явиться діалогове вікно (рис 3.6) підтвердження підключення, при натисканні «*Yes*» буде проведено з’єднання, при натисканні «*No*» відбудеться розрив з’єднання та видалення серверу.



а) б)

Рис. 3.5 – Випадаючий список

а) *Android* застосунку б) *Windows* застосунку

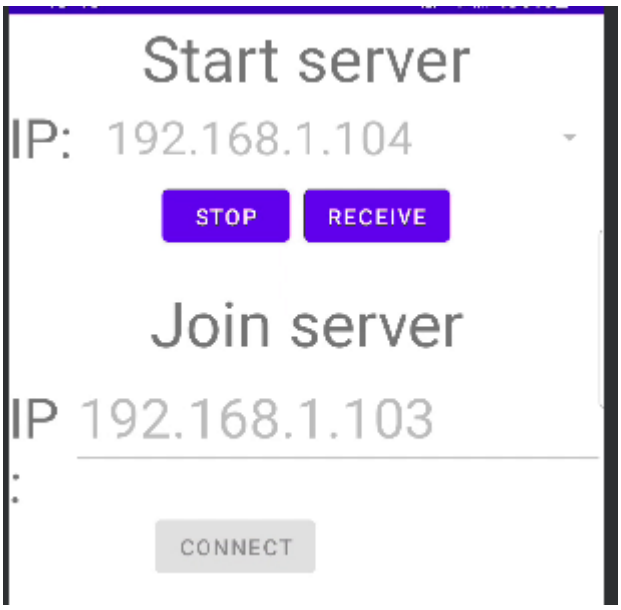


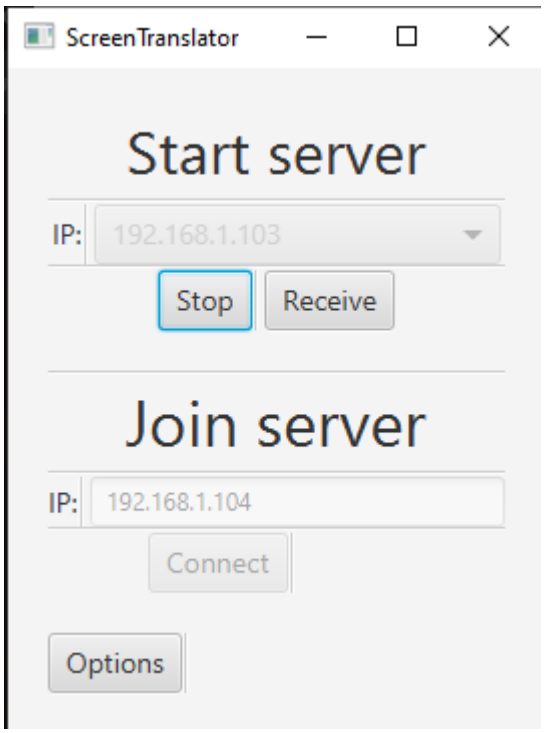
а) б)

Рис. 3.6 – Діалогове вікно підтвердження підключення

а) *Android* застосунку б) *Windows* застосунку

Якщо пристрої з’єдналися, то біля кнопки «*Stop*» з’явиться кнопка «*Receive*» (отримувати), а в полі під написом «*Join server*» (приєднатися до серверу), з’явиться адрес пристрою, що приєднався (рис. 3.7). Якщо користувач бажає то він може зупини процес з’єднання або розірвати з’єднання натиснувши на кнопку «*Stop*».





а) б)

Рис. 3.7 – Зміна головного вікна

а) *Android* застосунку б) *Windows* застосунку

Після натискання кнопки «*Receive*» відкривається вікно прийняття та відображення зображення екрану (рис. 3.8 та 3.9)

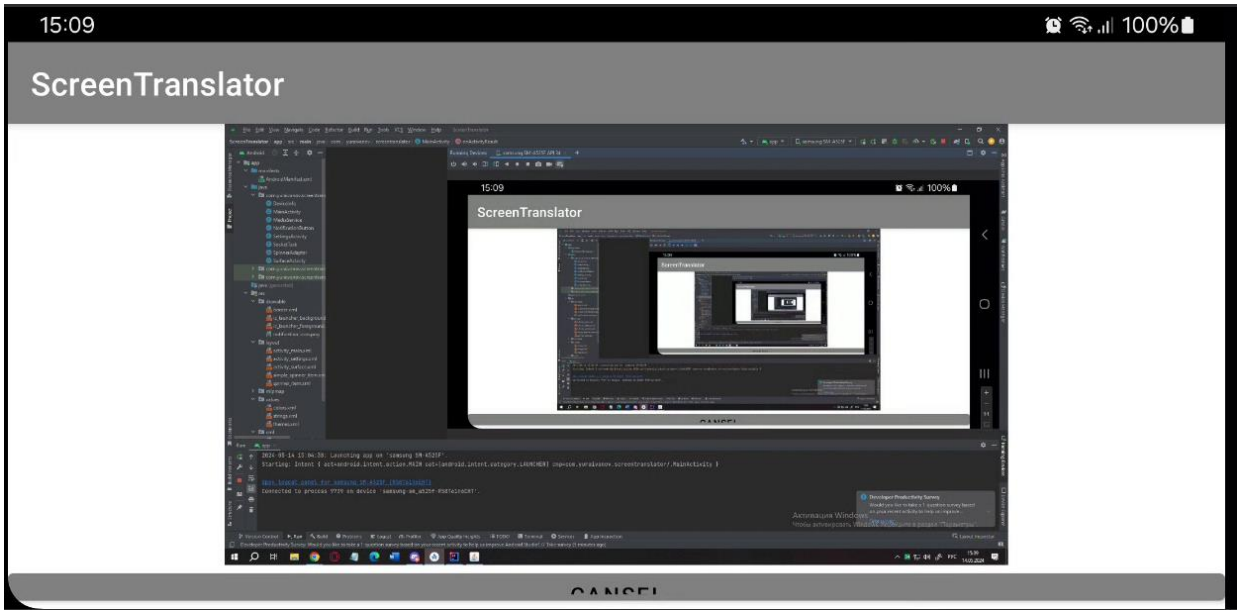


Рис. 3.8 – Вікно прийняття та відображення зображення екрану *Android* застосунку

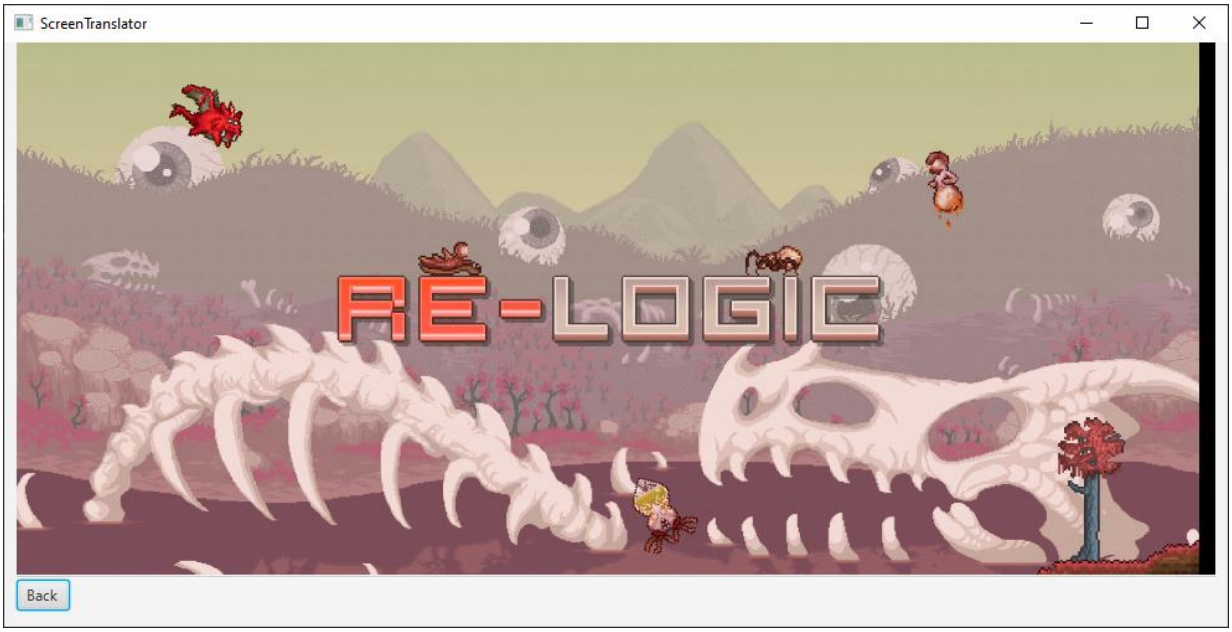
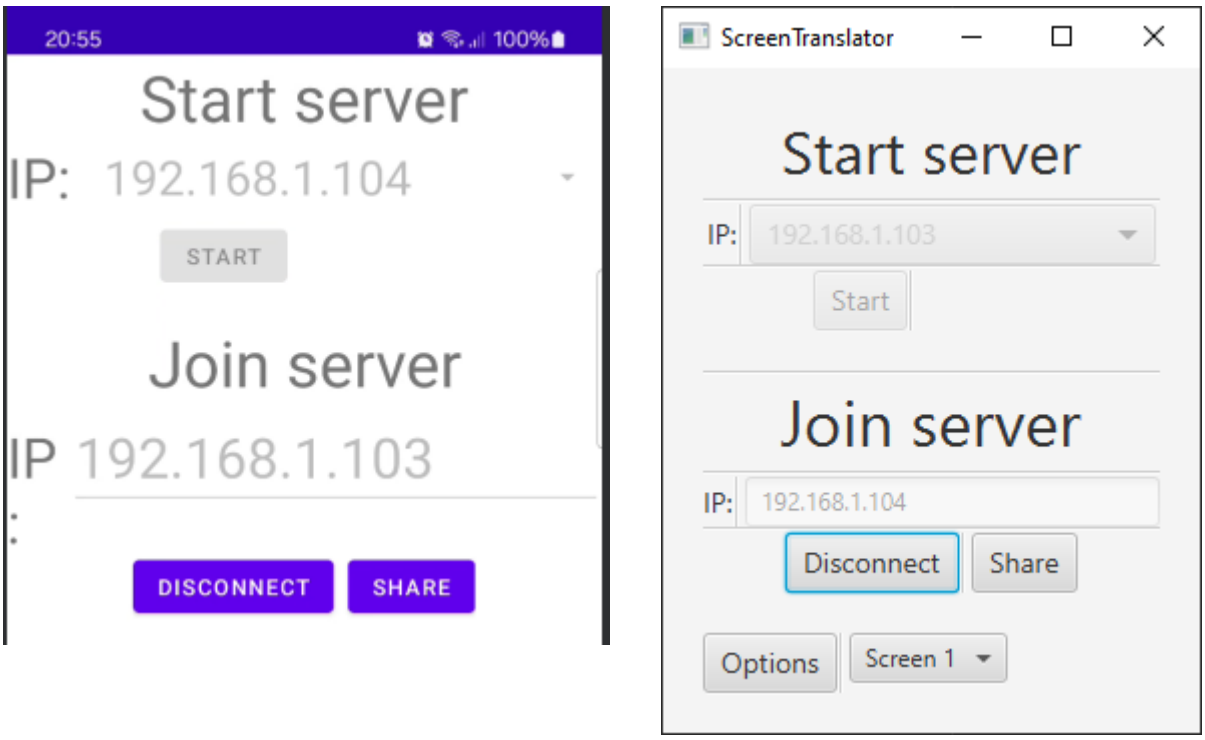


Рис. 3.9 – Вікно прийняття та відображення зображення екрану *Windows* застосунку

Щоб приєднатися до іншого пристрою користувач повинен ввести *IP*-адресу іншого пристрою в полі під написом «*Join server*», та натиснути кнопку «Connect» (приєднатися), кнопка змінне свою назву на «Disconnect» (від'єднатися). Якщо користувач бажає то він може зупини процес з'єднання або розірвати з'єднання натиснувши на кнопку «Disconnect».

Якщо пристрої з'єдналися, то біля кнопки «Disconnect» з'явиться кнопка «*Share*» (ділитися).

В даний момент дії застосунків на *Android* та *Windows* пристроях будуть розбігатися. В застосунку для *Android* з'являється лише кнопка «*Share*» (рис. 3.10 а), а в застосунку для *Windows* до кнопки ще додається випадаючий список з номерами підключених екранів (рис. 3.10 б).



а) б)

Рис. 3.10 – Зміна головного вікна

а) *Android* застосунку б) *Windows* застосунку

Після натискання кнопки «*Share*» в *Android* застосунку з'являється поле з попередженням об спробі запису або трансляції зображення екрану (рис. 3.11). Якщо користувач натисне «Скасувати» то захват та передача зображення не буде відбуватися, інакше якщо натисне «Почати зараз» то почнеться захват та передача зображення.

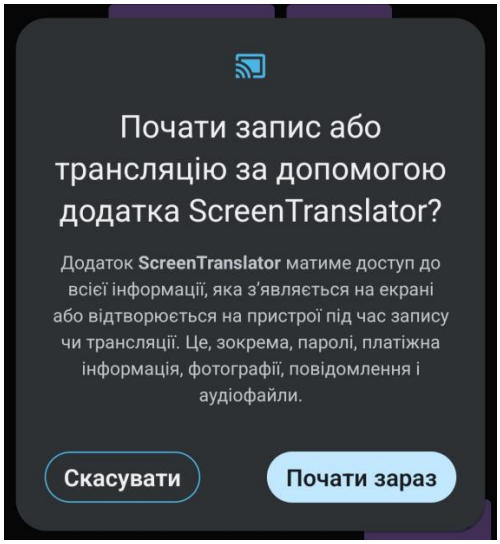


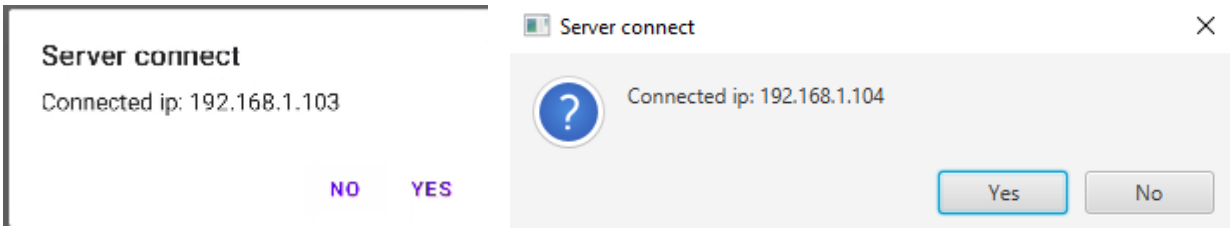
Рис. 3.11 – Попередженням об спробі запису або трансляції зображення екрану

В *Windows* застосунку перед натискання кнопки користувач повинен обрати екран, зображення якого він бажає передавати, і після натиснути кнопку «*Share*», ніяких додаткових попереджень, як в *Android* застосунку не виводиться.

Для зупинки трансляції користувач повинен натиснути кнопку «*Share*».

3.6 Реалізація безпеки

Як вже було сказано в пункті 2.4.3., при підключенні пристрою до серверу, на пристрою з сервером виводиться діалогове вікно (рис 3.12) підтвердження підключення.



а) б)

Рис. 3.12 – Діалогове вікно підтвердження підключення

а) *Android* застосунку б) *Windows* застосунку

А також в *Android* застосунку повинно виводиться вікно для підтвердження початку захвату екрану.

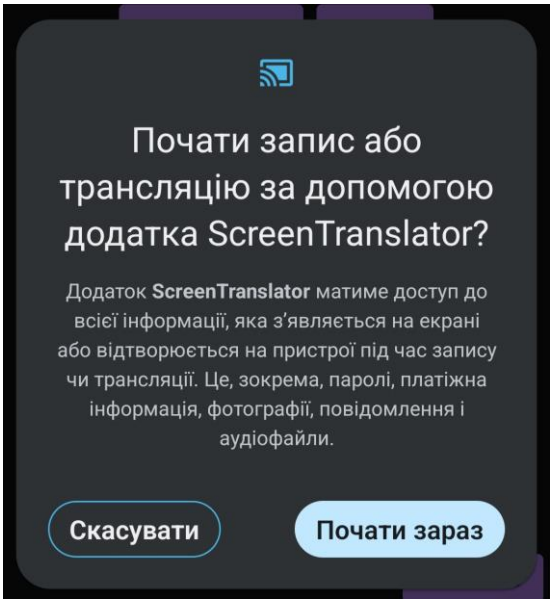


Рис. 3.13 – Попередженням об спробі запису або трансляції зображення екрану.

Висновок до третього розділу

У ході проведеної роботи було розроблено програмну та візуальну структуру для реалізації застосунку для системи передачі зображення екрану через мережу. Застосунки були запущені на декількох фізичних пристроях, ході його роботи та тестування функціональні потреби були виконані. Продемонстровано роботу та функціонал застосунків. Відображено процес з’єднання пристроїв та передачі зображення екрану.

РОЗДІЛ 4
ТЕХНІКО-ЕКОНОМІЧНА ЧАСТИНА

4.1 Економічне й маркетингове обґрунтування проекту.

У даній роботі проведено аналіз технологій передачі зображення екрана через мережу та розроблено програмне забезпечення передачі зображення екрана через мережу. Головною метою проекту було розробка «Системи передачі зображення екрана через мережу», яка дозволить користувачам ділитися візуальним контентом в реальному часі між різними пристроями.

У процесі розробки були використані різноманітні технології та інструменти. Зокрема, для написання програмного коду використовувалися редактори коду *Android Studio* для розробки на *Android* пристроїв та *IntelliJ IDEA* розробки на для персональних комп'ютерів, які надають зручні умови для розробки на мові програмування *Java*.

Перед початком розробки було проведено аналіз існуючих аналогів, визначено основні проблеми та завдання, які потрібно вирішити. Були досліджені моделі та методи проектування програмного забезпечення, а також проаналізовані технології передачі зображення екрана через мережу.

В результаті розроблено «Систему передачі зображення екрана через мережу», що може передавати зображення через локальну мережу та може використовуватися на різних платформах. Використання *Android Studio* та *IntelliJ IDEA* надає можливості для створення функціональних та зручних у використанні програм, а використання мови програмування *Java* забезпечує гнучкість та швидкість розробки. Завдяки підключенню пристроїв через вказання *IP*-адреса забезпечується незалежність від наявності Інтернет з'єднання мережі.

В таблиці 4.1 представлена порівняльна характеристика програм конкурентів та яка розробляється:

Таблиця 4.1

Порівняльна характеристика програм конкурентів

Особливості	*TeamViewer*	*RustDesk*	*Parsec*	*Chrome Remote Desktop*	*AnyDesk*	Screen-Translator
Платформи	*Windows*, *macOS*, *Linux*, *Android*, *iOS*	*Windows*, *macOS*, *Linux*	*Windows*, *macOS*, *Linux*	*Windows*, *macOS*, *Linux*, *Android*, *iOS*	*Windows*, *macOS*, *Linux*, *Android*, *iOS*	*Windows*, *Android*
Безпека	Шифрування *end-to-end*, двофакторна аутентифікація	Шифрування, *TLS*	*AES*-256 шифрування	Шифрування трафіку	Шифрування, *TLS*	-
Функції	Доступ до віддаленого робочого столу, передача файлів, відеоконференції	Віддалений робочий стіл, передача файлів	Віддалений робочий стіл, стрімінг відео, графіка	Доступ до віддаленого робочого столу	Віддалений робочий стіл, передача файлів	Доступ до зображення екрану пристроїв

Продовження таблиці 4.1

Особливості	*TeamViewer*	*RustDesk*	*Parsec*	*Chrome Remote Desktop*	*AnyDesk*	Screen-Translator
Відмінності	Комерційна ліцензія, інтеграція зі *Slack* та іншими сервісами	Відкритий код, безкоштовний	Гравець гри, стрімінг, розширені графічні можливості	Інтегровано з *Google*-акаунтом, безкоштовно	Швидкий доступ, оптимізовано для низької затримки	Предача даних лише через локальну мережу
Ціна	Безкоштовно для особистого використання, комерційні тарифи	Безкоштовно, преміум плани	Безкоштовно, преміум плани	Безкоштовно	Безкоштовно для особистого використання, комерційні тарифи	Безкоштовно

4.1.1 Організаційне обґрунтування

Таблиця 4.2

Класифікаційна оцінка проекту

клас	мультипроект
тип	змішаний
вид	комбінований
тривалість	короткостроковий
за степенем складності	проект средньої складності
рівень	галузевий

Мета – Розробка «Системи передачі зображення екрана через мережу», яка дозволить користувачам ділитися візуальним контентом в реальному часі між різними пристроями.

Результат – розроблений програмний продукт та інструкція по його використанню.

Етапи виконання розділів кваліфікаційної роботи:

– постановка технічного завдання: в цьому розділі представлені: аналіз існуючих систем та технологій, порівняння характеристик існуючих систем та технологій; вимоги до розробки програмного продукту; постановка задачі. Термін виконання – 1 місяць;

– розробка робочого проекту: на стадії розробки робочого проекту створюється вся необхідна документація об’єкта (специфікації, пояснювальна записка, інструкції та ін.). Термін виконання – 7 днів;

– техніко-економічна частина: проведення розрахунків собівартості та обґрунтування економічної доцільності впровадження даного програмного продукту. Термін виконання – 14 днів;

У кваліфікаційній роботі магістра представлений наступний склад робіт:

– технічне завдання. Термін виконання – 10-12 днів;

– розробка алгоритму програмного продукту. Термін виконання – 27-29 днів;

– розробка робочого проекту. Термін виконання – 30-35 днів;

– впровадження проекту. Термін виконання – 10-14 днів.

За структуру розроблювального проекту прийнята структура, орієнтована на результати проекту.

4.1.2 Склад робіт по життєвому циклу проекту

Таблиця 4.3

Життєвий цикл проекту

№ код роботи	Назва роботи	Т, дні
1-2	Збір та аналіз даних, існуючих моделей	10
2-3	Встановлення потреби в результатах	9
2-4	Затвердження концепцій	3
3-5	Визначення цілей, мотивів та вимог замовника та власника	11
4-5	Планування предметної області та інших елементів проекту	6
5-6	Розробка та затвердження зведеного плану	7
5-7	Організація виконання робіт	9
6-8	Інформаційний контроль за виконанням робіт	6
7-8	Детальне моделювання	15
8-9	Керівництво і координація робіт, корегування основних показників проекту	30
9-10	Підтвердження закінчення робіт	3

Продовження таблиці 4.3

№ код роботи	Назва роботи	Т, дні
9-11	Експлуатаційні випробування остаточного продукту проекту	5
10-12	Підготовка звітів	7
12-13	Оцінка результатів проекту	2
13-14	Підготовка керівництва з користування програмного продукту	3

Рис. 4.1 - Структура проекту

За складом робіт складений мережевий графік рис. 4.2.

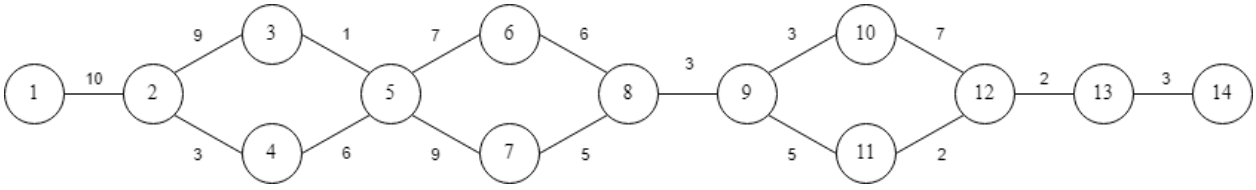


Рис. 4.2 - Мережевий графік проекту

4.1.3 Маркетингове обґрунтування проекту

Для створення програмного продукту були проаналізовані основні проблеми предметної області, проведений аналіз існуючих аналогів, проведено аналіз та обґрунтування обрання засобів реалізації, були досліджені моделі та методи проектування програмного забезпечення, були проаналізовані технологій передачі зображення екрана через мережу.

Створений програмний продукт зможе передавати зображення через локальну мережу інтернету завдяки підключенню пристроїв через їх *IP*-адресси, що робить її незалежною від наявності самого інтернету та безпечною для передаваємих даних. У якості середовища розробки обрано *Android Studio* для *Android* пристроїв та *IntelliJ IDEA* для персональних комп'ютерів, в якості мови програмування – *Java*.

Очікувані конкурентні переваги. Даний продукт не дивлячись на деякі недоліки, являє собою простий і зручний у застосуванні інтерфейс, проста реалізація функцій, що має забезпечити його затребуваність та актуальність.

Аналізуючи ринок в Україні та Сполучені Штати Америки (США) має суттєві відмінності, зокрема й у ціноутворенні. Ці відмінності можуть бути обумовлені різними факторами, такими як:

1. Рівень доходів – в середньому, доходи в США значно вищі, ніж в Україні. Це може призвести до того, що розробники та видавці ігор встановлюють вищі ціни на свої продукти в США, вважаючи, що споживачі там можуть собі це дозволити.

2. Податки та збори – різні країни мають різні податки та збори, які можуть впливати на кінцеву ціну ігор.

3. Валютні курси – курси валют можуть призвести до того, що ігри, які коштують однаково в одній валюті, матимуть різну ціну в іншій.

4. Конкуренція – рівень конкуренції на ринках України та США може відрізнятися. Це може впливати на ціноутворення, адже розробники та видавці можуть встановлювати нижчі ціни, щоб залучити більше покупців.

5. Піратство – рівень піратства в Україні вищий, ніж в США. Це може призвести до того, що розробники та видавці встановлюють вищі ціни на свої продукти в Україні, щоб компенсувати втрати від піратства.

Щоб дослідити ці відмінності, я порівняв ціни на декілька продуктів в Україні та США:

Таблиця 4.4

Продукт	Україна	США
TeamViewer for team	1,200 грн в місяць для 300 пристроїв та 15 користувачів	113 доларів в місяць для 300 пристроїв та 15 користувачів
Parsec for team	30 доларів в місяць з користувача	30 доларів в місяць з користувача

4.2 Економічні розрахунки проекту.

4.2.1 Визначення трудомісткості розробки

Тривалість розробки ПП залежить від обсягу інформаційної системи(ІС), трудомісткості її розробки, кваліфікації кадрів, а також планових термінів, що диктуються умовами ринку. У якості вихідних даних для визначення трудомісткості розробки ПП визначається обсяг програмних засобів в тисячах умовних машинних команд програми-аналога. Вибравши аналог програмного засобу (ПЗ), що містить V_0 в умовних машинних командах. У даному проекту розробляється новий програмний продукт, який відповідає аналогу ПЗ оптимізаційних розрахунків с V_0 = 3405 умовних машинних команд із трудомісткістю $T_р$ = 262 чол/год.

Трудомісткість розроблювального ПП визначається на кожному етапі окремо на підставі трудомісткості аналога з урахуванням складності розробки, ступеня новизни й ступеня використання в розробці стандартних модулів на підставі формул 4.1 – 4.4.

$$Т_{ТЗ} = Т_Р * L_1 * К_Н \quad (4.1)$$

$$Т_{ТП} = Т_Р * L_2 * К_Н \quad (4.2)$$

$$Т_{РП} = Т_Р * L_3 * К_Н * К_Т \quad (4.3)$$

$$Т_{ВН} = Т_Р * L_4 * К_Н \quad (4.4)$$

де $Т_р$ – укрупнення норма часу на розробку аналога ПЗ, чол/год, що коректується поправочним коефіцієнтом, що враховує умови розробки ПЗ, тобто в умовах комп'ютера, $К_н$=0.7, тобто:

$$Т_Р = 262 * 0.7 = 183{,}4 \text{ люд/год}$$

Даний проект можна віднести до ступеня новизни: В

Lj – питома вага *j*-го етапу розробки (залежно від ступеня новизни й відповідних стадій):

$$L_1 = 0{,}12;$$

$$L_2 = 0{,}11;$$

$$L_3 = 0{,}61;$$

$$L_4 = 0{,}16.$$

$К_Н$ – поправочний коефіцієнт, що враховує ступінь новизни, у цьому випадку $К_Н$ = 0,7; $К_Т$ – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм $К_Т$ = 0,8.

Тоді:

$$Т_{ТЗ} = 183{,}4 * 0{,}12 * 0{,}7 = 16 \text{ (люд/год)}$$

$$Т_{ТП} = 183{,}4 * 0{,}11 * 0{,}7 = 15 \text{ (люд/год)}$$

$$Т_{РП} = 183{,}4 * 0{,}61 * 0{,}7 * 0{,}8 = 55 \text{ (люд/год)}$$

$$Т_{ВН} = 183{,}4 * 0{,}16 * 0{,}7 = 21 \text{ (люд/год)}$$

Тривалість розробки ПП уроках визначається за формулою 4.5

$$Т_{пп} = \frac{\sum Т_{ij}}{8*0.73} \quad (4.5)$$

де $Т_{ПП}$ – сумарна тривалість розробки, розрахуємо:

$$Т_{пп} = \frac{104{,}91}{8*0.73} = 18 \text{ (дні)} = 0{,}049\text{(р.)}$$

4.2.2 Визначення ціни ПП

Оскільки ПП розглядається й створюється як продукція виробничо-технічного призначення, що допускає багаторазове тиражування й відчуження від безпосередніх розроблювачів, значить:

$$Ц = С * К + П_р \quad (4.6)$$

де:

С – витрати на розробку програмної продукції (кошторисна собівартість);

К – коефіцієнт обліку витрат на виготовлення досвідченого зразка ПП як продукції виробничо-технічного призначення;

$П_Р$ – нормативний прибуток, що розраховується по формулі:

$$П_Р=(С-С_М) * \frac{Р_н}{100} \quad (4.7)$$

де $Р_Н$ – норматив рентабельності, 25%;

$С_М$ – матеріальні витрати, грн./вироб.

Витрати на розробку програмної продукції можуть бути представлені у вигляді кошторису витрат, що включає в себе наступні статті:

1. Матеріали. Витрати на матеріали визначаються по формулі 4.8:

$$С_М = К_{МР} * \sum Ц_i * V_i \quad (4.8)$$

де $К_{МР}$ – коефіцієнт транспортно-заготівельних видатків;

$Ц_i$ – ціна одиниці *i*-го матеріалу, грн.;

V_i – придбана кількість *i*-го матеріалу.

В таблиці 4.5 представлено витрати на матеріали.

Таблиця 4.5

Витрати на матеріали

Найменування товару	Опис матеріалу	Кількість	Ціна за одиницю, грн.	Сума, грн.
Упаковка паперу	Упаковка офісного паперу А4	1	118	118
Флеш-накопичувач	Флеш память *USB Transcend JetFlash* 700 32*GB*	1	259	259
Картридж для принтеру	картридж *Canon i-Sensys MF*3010	1	300	300
Усього				677
$К_{МР}$ = 0,1				67,7
Разом:				744,7

2. Спеціальні устаткування.

Витрати, які пов’язані з використанням обчислювальної техніки, визначаються по формулі:

$$С_{ЕОМ} = t_{ЕОМ} * К_{И}^{ЕОМ} * Ц_{ЕОМ} * К_{Е}^{ЕОМ} * К_{БД}^{ЕОМ} \quad (4.9)$$

де $t_{ЕОМ}$– час використання ЕОМ для розробки даного ПО, год (104,91)

$К_{И}^{ЕОМ}$ – поправочний коефіцієнт обліку часу використання ЕОМ (1,08);

$Ц_{ЕОМ}$ – ціна 1-ої години роботи ЕОМ, грн. (7);

$К_{Е}^{ЕОМ}$ – 1,0;

$К_{БД}^{ЕОМ}$ – 1,0 (не використовується).

Тоді:

$$С_{ЕОМ} = 104,91 * 1,08 * 7 * 1 = 793,12 \text{ грн.}$$

3. Основна заробітна плата

У статтю включається основна заробітна плата одного виконавця, безпосередньо зайнятого розробкою даного ПП, з обліком його посадового

окладу (2880 грн. відповідно) і часу участі в розробці. Розрахунок ведеться по формулі 4.10:

$$С_{ЗО}= \frac{З*К_0*\tau}{Д_р} \quad (4.10)$$

де З – середньомісячний оклад виконавця, грн.;

$К_0$ – коефіцієнт обліку окладу керівників і консультантів проекту

τ – трудомісткість робіт, виконуваних виконавцем, люд/дні.

$Д_р$ – середня кількість робочих днів у місяці (22);

Тоді:

$$С_{ЗО}= \frac{2880*1*18}{22} = 2356{,}36 \text{ грн.}$$

4. Додаткова заробітна плата. Розрахунок по формулі 4.11:

$$С_{ЗД} = С_{ЗО} * К_д \quad (4.11)$$

де $К_Д$ – коефіцієнт відрахувань на додаткову заробітну плату (0,1).

$$С_{ЗД} = 2356{,}36 * 0{,}1 = 235{,}64 \text{ грн}$$

5. Відрахування на соціальне страхування.

У статті враховуються відрахування в бюджет соціального страхування по встановленому законодавством тарифу від суми основних й додаткової заробітної плати, тобто:

$$С_{СС} = К_{СС} * (С_{ЗО} + С_{ЗД}) \quad (4.12)$$

де $К_{СС}$ – коефіцієнт відрахувань на соціальне страхування (22%).

Тоді:

$$С_{СС} = 0{,}22 * (2356{,}36+ 235{,}64) = 570{,}24 \text{ грн}$$

6. Накладні витрати

У статті враховуються витрати на загальногосподарські витрати, позавиробничі (комерційні) витрати й витрати на керування. Накладні витрати визначають у відсотковому відношенні до основної заробітної плати, тобто:

$$С_Н = К_Н * С_{ЗО} \quad (4.13)$$

де $К_Н$ – коефіцієнт накладних видатків (50%).

Тоді:

$$С_Н = 0{,}5 * 2356{,}36 = 1178{,}18 \text{ грн.}$$

Результати розрахунку кошторисної вартості ПП можна продемонструвати у вигляді таблиці в табл. 4.6.

Таблиця 4.6

Результати розрахунку вартості

Найменування статті	Значення собівартості, грн.	Питома вага, %
Матеріали	744,7	5,55
Спеціальне устаткування	8325,5	62,08
Основна заробітна плата	2356,36	17,57
Додаткова заробітна плата	235,64	1,76
Відрахування на соціальне страхування	570,24	4,25
Накладні витрати	1178,18	8,79
Разом	5878,24	100

Тепер можемо розрахувати формули описані вище:

$$С = С_М + С_{ЕОМ} + С_{ЗО} + С_{ЗД} + С_{СС} + С_Н = 5878,24 \text{ грн.}$$

$$П_Р = (5878,24 – 744,7) * 0,25 = 1283,39 \text{ грн.}$$

$$Ц = 1,1 * 5878,24 + 1283,39 = 7749,45 \text{ грн.}$$

4.2.3 Визначення показника економічної ефективності

Очікуваний економічний ефект визначається за формулою:

$$Е_О = Е_Г – Е_Н * К_П \quad (4.14)$$

де $Е_Г$ – річна економія на поточних витратах (грн.);

$К_П$ – одноразові витрати на проект (грн.). У цьому випадку: вартість комп’ютера – 10 000 грн та ПП – 7749,45 грн;

E_H – нормативний коефіцієнт ефективності одноразових витрат (рекомендований $E_H = 0,25$; може бути також заданий господарюючим суб'єктом, або приймається на рівні процентної ставки по депозитних рахунках банку).

Річна економія складається з поточних витрат до і після впровадження ПП, у такий спосіб:

$$E_г = (C_1 – C_2) + \Delta П \quad (4.15)$$

де C_1, C_2 – відповідно поточні витрати, відповідно до й після впровадження проекту (грн.) і знаходяться за формолою 4.16;

ΔП – приріст прибутку господарюючого суб'єкта або його структурного підрозділу при впровадженні проекту (грн.) визначається експертним шляхом. В цьому випадку вона складе 0 грн.

$$C = C_{ЕОМ} + C_A + C_р + C_{всп} + C_{опл} \quad (4.16)$$

де $C_{ЕОМ}$ - витрати, пов'язані з використанням обчислювальної техніки, становлять:

$$C_{ЕОМ} = t_{ЕОМ} * К_И^{ЕОМ} * Ц_{ЕОМ} \quad (4.17)$$

де $t_{ЕОМ}$ – річний фонд часу роботи ЕОМ, який визначається виходячи з кількості робочих днів в році , тривалості робочого дня і з урахуванням часу на профілактичні огляди за рік :

$$t_{ЕОМ} = 8 * 365 = 2\,920 \text{ (год)}$$

$К_И^{ЕОМ}$ – поправочний коефіцієнт обліку часу використання ЕОМ (1,08);

$Ц_{ЕОМ}$ – ціна за 1 кВт світла, грн. (2,64);

$$C_{ЕОМ} = 2920 * 1,08 * 2,64 = 8\,325,5 \text{ грн}$$

C_A – сума річних амортизаційних відрахувань від вартості основного й допоміжного устаткування ІС (КМ) (25% від вартості устаткування);

$$C_A = 0.25 * 5878,24 = 1469,56 \text{ грн}$$

$C_р$ – вартість річного ремонту основного й допоміжного устаткування (6% $К_{ко}$);

$C_{всп}$ – річна вартість допоміжних матеріалів, пов'язаних з експлуатацією ІС (КМ) (2% $К_{ко}$);

$C_{опл}$ - основна і додаткова оплата праці персоналу

$$С_{опл} = З_{ЗО} + З_д + З_{сс} \quad (4.18)$$

де $З_{ЗО}$ – заробітна плата працівника за рік:

$$З_{ЗО} = 2880 * 12 = 34560 \text{ грн}$$

$З_д$ – додаткова заробітна плата:

$$З_д = 34560 * 0,1 = 3456 \text{ грн}$$

$З_{сс}$ – відрахування на соціальне страхування:

$$З_{сс} = (34560 + 3456) * 0,22 = 1520,64 \text{ грн}$$

Таким чином, становлять:

$$С_{опл} = 34560 + 3456 + 1520,64 = 39536,64 \text{ грн}$$

Так як ПП не потребує подальшого обслуговування то для $С_2$ $С_{опл}$ буде дорівнювати 0;

Одноразові витрати на проект складають:

$$КП = 10\,000 + 7749,45 = 17749,45 \text{ грн.}$$

Таким чином, поточні витрати ПП, становлять:

$$С_1 = 8\,325,5 + 3352,66 + 10\,000 * (0,06 + 0,02) + 39536,64 = 52014,8 \text{ грн}$$

$$С_2 = 8\,325,5 + 3352,66 + 10\,000 * (0,06 + 0,02) = 12478,16 \text{ грн}$$

$$Е_г = (52014,8 – 12478,16) + 0 = 39536,64 \text{ грн}$$

$$Е_о = 39536,64 – 0,25 * 17749,45 = 35099,28 \text{ грн}$$

Потім розраховується коефіцієнт ефективності одноразових витрат за формулою:

$$Е = \frac{Е_г}{К_п} \quad (4.19)$$

Якщо Е > Ен, то проект ефективний. Розрахуємо по формулі:

$$Е = \frac{39536,64}{17749,45} = 2,23$$

2,23 > 0,25 проект ефективний

Розраховується строк окупності одноразових витрат проекту, років:

$$Т = \frac{1}{Е} \quad (4.20)$$

Розрахуємо:

$$Т = \frac{1}{2,23} = 0,45$$

Основні економічні показники проекту надані в таблиці 4.7 [42]

Таблиця 4.7

Економічні показники проекту

Показники	Числове значення	Одиниці виміру
Тривалість розробки	18	Дні
Ціна ПП	7749,45	Грн
Капітальні затрати	17 749	Грн
Річна економія на поточних витратах після введення ПП	39 537	Грн
Економічний ефект	35 099	Грн
Коефіцієнт ефективності	2,23	–
Термін окупності проекту	0,45	Рік

4.3 Бізнес план стартап-проекту

Для успішного запуску та розвитку стартапу зі створення системи передачі зображення екрану в локальній мережі, потрібно розробити докладний бізнес-план. У ньому розглянуто всі ключові аспекти проекту, від аналізу ринку та опису продукту до фінансового плану та оцінки ризиків. Нижче наведено таблицю з основними пунктами бізнес-плану.

Таблиця 4.8

Бізнес план стартап–проекту

Ключові пункти	Опис
Основна ціль проекту	Створити інноваційну систему передачі зображення екрану через мережу для спрощення спільної роботи, навчання та демонстрації контенту в офісах, навчальних закладах та інших організаціях.

Продовження таблиці 4.8

Ключові пункти	Опис
Аналіз ринку	Підвищений попит в офісних та навчальних середовищах: існує високий попит на рішення для зручної та ефективної передачі зображення екрану усередині будівель та офісів. Обмежена конкуренція: незважаючи на наявність деяких рішень, більшість з них мають обмежені можливості або потребують складного налаштування.
Опис продукту	Моя система надаватиме такі функції: – просте налаштування та використання в локальній мережі. – висока якість передачі зображення без затримок. – підтримка різних пристроїв та операційних систем.
Бізнес модель	Продаж ліцензій: запропонуємо ліцензії на використання нашого програмного забезпечення для компаній та навчальних закладів. Додаткові послуги: можемо надати послуги консультації, налаштування та підтримки для корпоративних клієнтів.
Маркетинг та реклама	Прямий продаж: встановлення контактів з потенційними клієнтами через прямі продажі та персоналізовані пропозиції. Участь у виставках та конференціях: демонстрація продукту на галузевих заходах для привернення уваги до нього. Реклама у спеціалізованих виданнях: розміщення оголошень та статей про продукт у журналах та онлайн-ресурсах для офісного обладнання.

Продовження таблиці 4.8

Ключові пункти	Опис
Фінансовий план	Інвестиції: початкові інвестиції у розробку та маркетинг продукту. Прогнозовані доходи: продаж ліцензій та додаткові послуги. Витрати: витрати розробку, маркетинг, адміністративні витрати тощо. Окупність Плануємо досягти окупності протягом першого року після запуску.
Ризики	Технічні проблеми: необхідність забезпечити стабільну роботу продукту у всіх умовах мережі та на різних пристроях. Конкуренція: реагування на конкурентні продукти та пошук способів виділитися на ринку. Законодавчі обмеження: дотримання законодавства про захист даних та конфіденційності.
Висновки	Мій стартап пропонує рішення для передачі зображення екрану в локальній мережі з високою якістю, простотою використання та надійним захистом даних.

Висновки до четвертого розділу

Розробка даної системи передачі зображення екрану через мережу дозволяє значно знизити витрати на обладнання, оскільки використовуються вже існуючі пристрої. Це робить проект економічно вигідним та доступним для широкого кола користувачів. З маркетингової точки зору, високий попит на рішення для віддаленого управління та спільної роботи створює значні перспективи для комерційного успіху продукту. Система відповідає сучасним вимогам ринку, забезпечуючи ефективну та реалістичну передачу зображення екрану в режимі реального часу.

Результати економічних розрахунків підтверджують економічну доцільність пропонованого програмного продукту. Знижуються експлуатаційні витрати за рахунок зростання продуктивності праці, економії на поточних витратах за рік 39536,64 грн. Термін окупності становить 5,5 місяців, що є

прийнятним результатом і доводить економічну доцільність впровадження даного проекту.

Для успішного запуску та розвитку стартапу зі створення системи передачі зображення екрану в локальній мережі розроблено докладний бізнес-план, розглянуто всі ключові аспекти проекту, від аналізу ринку та опису продукту до фінансового плану та оцінки ризиків.

РОЗДІЛ 5
ОХОРОНА ПРАЦІ

5.1 Небезпечні та шкідливі речовини та фактори

Небезпечні фактори це ті, які становлять загрозу для життя чи здоров'я людини. Прикладом небезпечного фактору є змінний електричний струм напруги 220 В. Комп’ютер, як і будь-який електричний прилад, особливо при його неправильному підключенні, може бути джерелом ураження користувача ПК електричним струмом.

Шкідливі фактори, це ті фактори які здатні викликати професійні захворювання, або взагалі недуги. Шкідливими факторами при роботі з персональним комп’ютером є неіонізуюче випромінювання промислової частоти, збільшене нервово - емоційне навантаження на оператора, збільшення навантаження на органи зору та дрібні стереостатичні рухи кінцівок.

Згідно документу [43] шкідливими виробничими факторами являються:

– наявність шуму та вібрації;

– м’яке рентгенівське випромінювання;

– електромагнітне випромінювання;

– ультрафіолетове і інфрачервоне випромінювання;

– електростатичне поле між екраном і оператором;

– відсутність або недолік природного світла

– пряма та відбита блисткість

– наявність пилу, озону, оксидів азоту й аероіонізації

– показники мікроклімату: температура повітря T, відносна вологість W, швидкість руху повітря V.

Ці фактори можуть викликати у працівника певні розлади здоров’я, зокрема підвищення артеріального тиску, кон’юктивіти, тендовагініти та інші захворювання.

5.2 Класифікація приміщень за ступенем небезпеки

По ступеню пожежної небезпеки дане приміщення із ВДТ ЕОМ відноситься до категорії В за НАПБ Б.03.002-2007 - це приміщення, в яких є горючі гази (ГГ), легкозаймисті, горючі і важкогорючі рідини, а також речовини та матеріали, які здатні при взаємодії з водою, киснем повітря або один з одним вибухати і горіти або тільки горіти; горючий пил і волокна, тверді горючі та важкогорючі речовини і матеріали, за умови, що приміщення, в яких вони знаходяться (обертаються), не відносяться до категорій А, Б і питома пожежна навантага для твердих і рідких легкозаймистих та горючих речовин на окремих ділянках1площею не менше 10 м2 кожна перевищує 180 МДж/м. [44]

5.3 Об’ємно-планувальні рішення щодо розміщення обладнання

Розміщення робочих місць ВДТ ЭВМ в підвальних і цокольних поверхах забороняється. Робоче місце знаходиться у комп’ютерному класі. Приміщення знаходиться на першому поверсі 3-поверхової будівлі Приміщення з ВДТ має природне і штучне освітлення. Природне освітлення здійснюється через світлові отвори, орієнтовані переважно на північ, північний захід. На робочих місцях з ВДТ КПО (коефіцієнт природного освітлення) складає понад 1.5%. Приміщення з ВДТ обладнане системами опалювання, вентиляції і кондиціонування повітря. Віконні отвори обладнані регульованими пристосуваннями: це жалюзі, фіранки і зовнішні козирки.

Задане приміщення розміром:

– довжина - А = 12 м;

– ширина - В = 15 м;

– висота - Н =3,2 м.

Площа приміщень, в яких розташовують персональні комп’ютери (ПК), визначають згідно з діючими нормативними документами з розрахунку на одне робоче місце: площа - не менше 6,0 м2, об'єм - не менше 20,0 м3, з урахуванням максимальної кількості осіб, що одночасно працюють в зміні. У

нашому випадку площа - 180 м2, а об'єм – 576 м3. З цього виходить, що максимальна кількість робочих місць рівна двадцять вісім. Фактично в приміщені працює 16 чоловік

5.4 Електробезпека обладнання

Лінія електромережі для ЕОМ виконана як окрема групова трьохпровідна мережа, шляхом прокладки фазового, нульового робочого, нульового захисного проводів. Нульовий захисний провід використається тільки для занулення електроприймача. Використання нульового робочого проводу в якості нульового захисного заборонено. Штепсельні з'єднання й розетки мають спеціальні контакти для підключення нульового захисного проводу.

ПЕОМ не підключено до звичайної двохпровідної мережі, у тому числі з використанням перехідних пристосувань. Експлуатація кабелів і проводів з ушкодженою ізоляцією, саморобних подовжувачів не проводиться. [45]

Згідно з ППЕ НПАОП 40.1-1.32-01, всі приміщення за ступенем небезпеки ураження електричним струмом діляться на три класи:

– приміщення без підвищеної небезпеки;

– приміщення з підвищеною небезпекою;

– особливо небезпечні приміщення.

Приміщення з ПК відноситься до приміщення з підвищеною небезпекою, тому що існує можливість одночасного дотику людини до будь яких з'єднання з землею, металоконструкціями будинків, технологічними апаратами, механізмами і т.п., з одного боку, і до металевих корпусів електрообладнання з іншого.

Згілно з ППЕ, все електрообладнання ділиться на два класи - з напругою до 1000 В, і з напругою понад 1000 В. Електрообладнання приміщень з ПК відноситься до першого класу, з напругою до 1000 В.

ЕОМ, периферійні пристрої та інше обладнання (апарати управління світильники і тому подібне), електропроводи і кабелі по виконанню і ступеню захисту повинні відповідати класу зони по ППЕ.

Лінія електромережі для живлення ЕОМ, периферійних пристроїв ЕОМ і устаткування для обслуговування, ремонту і наладки ЕОМ виконана як окрема групова трьохдротова мережа.

5.5 Пожежна безпека

Приміщення, в якому розташовані ПК, має першу ступінь вогнестійкості згідно [46]. 1 ступінь вогнестійкості – це будинки з несучими та огороджувальними конструкціями з природних матеріалів або штучного каменю, бетону або залізобетону із застосуванням листових і плиткових негорючих матеріалів.

Приміщення з ПК оснащене автоматичними пожежними димовими сповіщувачами згідно з [47]. Як засоби пожежогасіння на даному об'єкті застосовуються вуглекислотні вогнегасники, призначені для гасіння спалахів установок напругою до 1000В. Приміщення категорії В з площею 180 кв.м має 4 вуглекислотні вогнегасника с заряд вогнегасної речовини 6 кг.

5.6 Виробнича санітарія

Приміщення з ПК матє загальне рівномірне освітлення. При роботі з документами загальне рівномірне доповнене місцевим. Джерелами світла для загального рівномірного освітлення є газорозрядні лампи низького тиску, а для місцевого дозволяється використовувати лампи розжарювання .

При роботі за дисплеєм освітленість визначається мінімальним об'єктом розрізнення шириною лінії рукописного або друкарського тексту, який читає користувач з листа. Освітлення на робочому місці становить 425 Лк [48].

5.7 Створення безпечного та комфортного робочого місця

Створення безпечного та комфортного робочого місця потребує комплексного підходу для усунення чи зменшення шкідливих факторів до нормативних порогів. Це включає оцінку ризиків, впровадження технічних та організаційних заходів, а також постійний моніторинг умов праці. Ось кілька кроків, які можна зробити для досягнення цієї мети:

1. Перш ніж приступати до усунення шкідливих факторів, необхідно провести оцінку ризиків:

– аналіз робочого місця: визначення та оцінка всіх можливих шкідливих факторів (наприклад, шум, вібрація, освітлення, хімічні речовини, електромагнітні поля).

– консультації зі співробітниками: проведення опитувань та розмов з працівниками для виявлення їх думок щодо потенційних ризиків та проблем на робочому місці.

– вимірювання та моніторинг: використання спеціалізованого обладнання для вимірювання рівнів шуму, вібрації, освітленості та інших факторів.

2. Технічні заходи спрямовані на усунення або мінімізацію впливу шкідливих факторів у джерелі:

– поліпшення вентиляції: забезпечує ефективну вентиляцію для зниження концентрації шкідливих речовин у повітрі;

– шумоізоляція: установка шумопоглинаючих матеріалів та конструкцій для зниження рівня шуму на робочому місці;

– усунення вібрації: використання антивібраційних опор та демпфуючих матеріалів для зниження рівня вібрації обладнання.

– оптимізація освітлення: встановлення регульованого освітлення, що відповідає стандартам різних типів робіт (наприклад, робота за комп'ютером, ручна праця);

– екранування та захист: встановлення екранів та захисних бар'єрів для зменшення впливу електромагнітних полів та інших фізичних факторів.

3. Організаційні заходи включають зміну процесів та методів роботи для зниження впливу шкідливих факторів:

– ротація та перерви: введення систем ротації працівників та регулярних перерв для зменшення впливу шкідливих факторів (наприклад, шум, вібрація);

– графіки роботи: оптимізація робочих графіків для запобігання перевтомі та зниження стресу;

– навчання та інформування: проведення регулярних інструктажів та тренінгів з питань охорони праці та безпеки, навчання правильному використанню обладнання та засобів індивідуального захисту.

– контроль та нагляд: призначення відповідальних осіб за контроль виконання заходів з охорони праці та регулярне проведення перевірок умов праці.

4. Використання ЗІЗ дозволяє захистити працівників від впливу шкідливих факторів:

– захист органів слуху: застосування навушників або берушів в умовах підвищеного шуму;

– захист органів дихання: використання респіраторів та масок при роботі з хімічними речовинами або у запилених приміщеннях;

– захист очей: носити захисні окуляри або екрани під час роботи з небезпечними матеріалами або обладнанням;

– захист рук та шкіри: застосування захисних рукавичок, кремів та інших засобів при роботі з хімікатами або в умовах підвищеного механічного навантаження.

5. Постійний моніторинг умов праці та регулярне оновлення заходів з охорони праці сприяють підтримці безпеки на робочому місці:

– регулярні вимірювання: періодичне проведення вимірювань рівнів шуму, освітлення, концентрації шкідливих речовин та інших факторів;

– аналіз інцидентів: розбір та аналіз нещасних випадків та інцидентів на робочому місці для виявлення причин та розробки заходів щодо їх запобігання у майбутньому;

– зворотній зв'язок: збір та аналіз відгуків працівників про заходи щодо покращення умов праці та внесення необхідних коригувань.

Висновок до п’ятого розділу

У ході розробки цієї частини дипломної роботи було розглянуто важливість дотримання вимог до охорони праці при створенні застосунків для системи передачі зображення екрану.

Було проаналізовано питання електробезпеки. При використанні електронного обладнання важливо дотримуватися правил безпечної експлуатації, зокрема, не користуватися пошкодженими електричними кабелями і вимикачами.

Таким чином, за результатами проведеної роботи, я маю усвідомлення про важливість дотримання норм охорони праці та промислової санітарії під час виконання дипломного проекту. Впровадження цих рекомендацій допоможе забезпечити безпечні та продуктивні умови для продовження роботи над моїм проектом.

ЗАГАЛЬНІ ВИСНОВКИ

За результатами аналізу предметної області, було виявлено загальні риси застосунків для систем передачі зображення екрану через мережу. Проаналізовані сучасні аналогічні застосунки з виявленням переваг та їх недоліків. Здійснено постановку задачі із зазначенням бажаного результату, щодо того, що є необхідним для створення застосунку такого типу. Під час проектування розроблено архітектуру застосунку. В ході проведеної роботи розроблено програмну та візуальну складові застосунку. Дана розробка проекту є вигідною за висновками того, що вона має перспективи прибутку та доволі швидку окупність. У ході розробки цієї частини дипломної роботи було розглянуто важливість дотримання вимог до охорони праці при створенні застосунку. Продемонстровано функціонал роботи додатку та різних його частин.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Was ist TeamViewer? [Веб-сайт] URL: https://www.heise.de/tipps-tricks/Was-ist-TeamViewer-4507319.html (дата звернення: 30.11.2023).

2. Die besten Programme für den USB-Stick – TeamViewer Portable [Веб-сайт] URL: https://www.t-online.de/digital/aktuelles/id_54675798/bilder/die-besten-programme-fuer-den-usb-stick-.html (дата звернення: 30.11.2023).

3. PC-Fernsteuerung mit RustDesk [Веб-сайт] URL: https://gnulinux.ch/pc-fernsteuerung-mit-rustdesk (дата звернення: 30.11.2023).

4. RustDesk | Open source remote desktop software | The Open Source TeamViewer Alternative [Веб-сайт] URL: https://rustdesk.com/ (дата звернення: 30.11.2023).

5. Low-latency desktop capture software Parsec is incredibly promising [Веб-сайт] URL: https://www.pcpowerplay.com.au/feature/lowlatency-desktop-capture-software-parsec-is-incredibly-promising,472886 (дата звернення: 30.11.2023).

6. Parsec Warp [Веб-сайт] URL: https://parsec.app/parsec.app (дата звернення: 30.11.2023).

7. Chromoting Build Instructions [Веб-сайт] URL: https://chromium.googlesource.com/chromium/src/+/refs/tags/82.0.4058.2/docs/old_chromoting_build_instructions.md (дата звернення: 30.11.2023).

8. Understanding the Remote Desktop Protocol (RDP) [Веб-сайт] URL: https://learn.microsoft.com/en-us/troubleshoot/windows-server/remote/understanding-remote-desktop-protocol (дата звернення: 30.11.2023).

9. Innovative and Reliable: Our Features [Веб-сайт]. URL: https://anydesk.com/en/features (дата звернення: 30.11.2023).

10. Aussies have lost over AU$7 million to remote access scams already this year [Веб-сайт]. URL: https://www.zdnet.com/article/aussies-have-lost-over-au7-million-to-remote-access-scams-already-this-year/ (дата звернення: 30.11.2023).

11. September iPod Event: In iOS 4.2, AirTunes Becomes AirPlay [Веб-сайт]. URL: http://www.cultofmac.com/september-ipod-event-in-ios-4-2-airtunes-becomes-airplay/57429 (дата звернення: 30.11.2023).

12. About AirPlay Mirroring in OS X Mountain Lion [Веб-сайт]. URL: http://support.apple.com/kb/HT5404 (дата звернення: 28.01.2024).

13. Google Launches the $35 Chromecast Streaming Device to Bring Chrome to the Living Room [Веб-сайт]. URL: https://techcrunch.com/2013/07/24/google-chromecast/ (дата звернення: 28.01.2024).

14. Google Opens Up Chromecast to Developers, Releases Cast SDK [Веб-сайт]. URL: https://thenextweb.com/news/google-opens-chromecast-developers-releases-cast-software-development-kit-apps-websites (дата звернення: 28.01.2024).

15. Google Quietly Phases Out 'Google Cast' Branding for TVs, Speakers [Веб-сайт]. URL: https://variety.com/2016/digital/news/google-cast-becoming-chromecast-1201925162/ (дата звернення: 28.01.2024).

16. On Wifi-Display, Democratic Republics and Miracles [Веб-сайт]. URL: https://dvdhrm.wordpress.com/2014/02/17/on-wifi-democratic-republics-and-miracles/ (дата звернення: 28.01.2024).

17. Samsung screen mirroring finally supports casting to a Chromecast [Веб-сайт]. URL: https://www.androidpolice.com/samsung-google-cast/ (дата звернення: 28.01.2024).

18. Wireless Display Standards Explained: AirPlay, Miracast, WiDi, Chromecast, and DLNA [Веб-сайт]. URL: https://www.howtogeek.com/177145/wireless-display-standards-explained-airplay-miracast-widi-chromecast-and-dlna/ (дата звернення: 28.01.2024).

19. What formats does Miracast support? [Веб-сайт]. URL: https://www.wi-fi.org/knowledge-center/faq/what-formats-does-miracast-support (дата звернення: 28.01.2024).

20. Miracast unter Linux ist schrecklich [Веб-сайт]. URL: https://www.golem.de/sonstiges/zustimmung/auswahl.html?from=https%3A%2F%

2Fwww.golem.de%2Fnews%2Fwifi-display-miracast-unter-linux-ist-schrecklich-1402-104315.html&referer=https%3A%2F%2Fen.wikipedia.org%2F (дата звернення: 28.01.2024).

21. Understanding Remote Desktop Protocol (RDP) – Windows Serve [Веб-сайт]. URL: https://learn.microsoft.com/en-us/troubleshoot/windows-server/remote/understanding-remote-desktop-protocol (дата звернення: 28.01.2024).

22. How to change the listening port for Remote Desktop [Веб-сайт]. URL: https://learn.microsoft.com/ru-RU/windows-server/remote/remote-desktop-services/clients/change-listening-port (дата звернення: 28.01.2024).

23. Service Name and Transport Protocol Port Number Registry [Веб-сайт]. URL: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml?&page=64 (дата звернення: 28.01.2024).

24. rdesktop: A Remote Desktop Protocol Client [Веб-сайт]. URL: http://www.rdesktop.org/#docs (дата звернення: 28.01.2024).

25. Virtual network computing [Веб-сайт]. URL: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://quentinsf.com/publications/virtual-network-computing/vnc-ieee.pdf (дата звернення: 28.01.2024).

26. The VNC family of Remote Control Applications: a list of VNC variants [Веб-сайт]. URL: https://ipinfo.info/html/vnc_remote_control.php (дата звернення: 28.01.2024).

27. TCP/IP Overview — Cisco [Веб-сайт]. URL: http://www.cisco.com/c/en/us/support/docs/ip/routing-information-protocol-rip/13769-5.html (дата звернення: 28.01.2024).

28. IP Application Services Configuration Guide, Cisco IOS XE Release 3S — Configuring TCP [Cisco IOS XE 3S] — Cisco [Веб-сайт]. URL: http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipapp/configuration/xe-3s/iap-xe-3s-book/iap-tcp.html (дата звернення: 28.01.2024).

29. RFC 6335 — Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Transport Protocol Port Number and Service Name Registry [Веб-сайт]. URL: https://tools.ietf.org/html/rfc6335

30. Xbox 360 Network Ports and Router Configurations for Xbox Live [Веб-сайт]. URL: http://support.xbox.com/en-US/xbox-360/networking/network-ports-used-xbox-live (дата звернення: 28.01.2024).

31. Kurose J.F. , Ross K.W. Computer Networking: A Top-Down Approach (5th Edition). Boston, 2010. 864 p.

32. Clark, M.P. Data Networks IP and the Internet (1th Edition.). England, 2003. 867 p.

33. UDP Protocol Overview [Веб-сайт]. URL: https://ipv6.com/articles/general/User-Datagram-Protocol.htm (дата звернення: 28.01.2024).

34. Forouzan, B.A. (2000). TCP/IP: Protocol Suite, 1st ed. New Delhi, India: Tata McGraw-Hill Publishing Company Limited. (дата звернення: 28.01.2024).

35. The Java Language Environment [Веб-сайт]. URL: http://www.oracle.com/technetwork/java/intro-141325.html (дата звернення: 15.02.2024).

36. Get started with Kotlin [Веб-сайт]. URL: https://kotlinlang.org/docs/getting-started.html (дата звернення: 15.02.2024).

37. Kotlin Multiplatform [Веб-сайт]. URL: https://kotlinlang.org/docs/multiplatform.html (дата звернення: 15.02.2024).

38. Evolving a language in and for the real world: C++ 1991—2006, Bjarne Stroustrup [Веб-сайт]. URL: https://web.archive.org/web/20071120015600/http://www.research.att.com/~bs/hopl-almost-final.pdf (дата звернення: 15.02.2024).

39. Welcome to C# 11 [Веб-сайт]. URL: https://devblogs.microsoft.com/dotnet/welcome-to-csharp-11/ (дата звернення: 15.02.2024).

40. Meet Android Studio [Веб-сайт]. URL: https://developer.android.com/studio/intro (дата звернення: 15.02.2024).

41. IntelliJ IDEA Features and Editions Comparison [Веб-сайт]. URL: http://www.jetbrains.com/idea/features/ant_maven.html (дата звернення: 15.02.2024).

42. Методичні вказівки до оцінки науково-технічної ефективності розробки нової технології, нового обладнання та інших інновацій. Для студентів всіх спеціальностей СВО «бакалавр» і «магістр» денної і заочної форм навчання. Укладачі Басюркіна Н.Й., Свистун Т.В. Одеса: ОНТУ, 2022 р. 18 с.

43. ГОСТ 12.0.003-74. Система стандартів безпеки праці. Небезпечні та шкідливі виробничі фактори. Класифікація. - Введ. 18.11.74. - М.: Вид-во стандартів, 1974. - 4 с.

44. ДСТУ Б В.1.1-36:2016 Визначення категорій приміщень, будинків, установок за вибухопожежною та пожежною небезпекою,. – К.: МІНРЕГІОНБУД України, 2016. – 66 с.

45. НПАОП 40.1-1.32-01. Правила будівництва електроустановок. Електрообладнання спеціальних установок.

46. ДБН В.1.1.7-2016 Пожежна безпека обєктів будівництва.Загальні вимоги.

47. ДБН В 2.5-56-2010 Системи противопожежного захисту. Київ 2015-с.134.

48. ДБН В.2.5-28-2019 Інженерне обладнання будинків і споруд. Природне і штучне освітлення.