Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Розробка додатку відеотрансляції під мобільні пристрої на базі операційної системи Android

Виконав: студент VI курсу, групи СНм-61
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

_____ Бондаренко В.С.
(підпис) (прізвище та ініціали)

Керівник _____ Готович В.А.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Дуда О.М.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Загородна Н.В.
(підпис) (прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«25» _грудня 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Бондаренко Владислав Сергійович
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка додатку відеотрансляції під мобільні пристрої на базі операційної системи Android

Керівник роботи Готович Володимир Анатолійович, к.т.н., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 29 » листопада 2024 року № 4/7-1141

2. Термін подання студентом завершеної роботи 7 грудня 2024 р.

3. Вихідні дані до роботи Наукові публікації про системи відеотрансляції для операційної Системи Android

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Аналіз вимог та предметної області
2 Проектування програмної системи
3. Розробка та тестування програмного додатку
4 Охорона праці та безпека в надзвичайних ситуаціях.
Висновки.
Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титульний слайд 2. Слайд з Актуальністю 3. Основні задачі дослідження 4. Слайд Порівняння аналогів 5. Слайд порівняння WebRTC і RTSP 6. Слайд вимог до програми 7. Слайд з поясненням архітектури і взаємодії 8. Огляд функціоналу клієнту Android. 9. Огляд сигнального серверу 10. Огляд клієнту пк і опис функціоналу 11. Висновок

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Сенчишин В.С., доцент		
Безпека в надзвичайних ситуаціях	Клепчик В.М., ст. викладач		

7. Дата видачі завдання 29 листопада 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	29.11.2024	*Виконано*
2.	Підбір наукових джерел про системи відеотрансляції	29.11.2024-04.12.2024	*Виконано*
	на базі операційної системи Android		
3.	Опрацювання наукових публікацій та збір даних по	29.11.2024-04.12.2024	*Виконано*
	темі роботи		
4.	Виконання дослідження згідно мети кваліфікаційної	05.12.2024-09.12.2024	*Виконано*
	Роботи		
5.	Оформлення розділу «Аналіз вимог та предметної області»	05.12.2024-09.12.2024	*Виконано*
6.	Оформлення розділу «Проектування програмної системи»	05.12.2024-09.12.2024	*Виконано*
7.	Оформлення розділу «Розробка та тестування програмного додатку.»	05.12.2024-09.12.2024	*Виконано*
8.	Виконання завдання до підрозділу «Охорона праці»	02.12.2024-09.12.2024	*Виконано*
9.	Виконання завдання до підрозділу «Безпека в	02.12.2024-09.12.2024	*Виконано*
	надзвичайних ситуаціях»		
10.	Оформлення кваліфікаційної роботи	10.12.2024-11.12.2024	*Виконано*
11.	Нормоконтроль	12.12.2024-13.12.2024	*Виконано*
12.	Перевірка на плагіат	16.12.2024	*Виконано*
13.	Попередній захист кваліфікаційної роботи	17.12.2024	*Виконано*
14.	Захист кваліфікаційної роботи	26.12.2024	

Студент _____ Бондаренко В.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Готович В.А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка додатку відеотрансляції під мобільні пристрої на базі операційної системи Android // Кваліфікаційна робота освітнього рівня «Магістр» // Бондаренко Владислав Сергійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНм-61 // Тернопіль, 2024 // С. 78, рис. – 14, додат. – 2, бібліогр – 20

Ключові слова: відеоспостереження, kotlin, python, kivy, webrtc, rtsp, клієнт-серверна архітектура, реальний час, безпека, моніторинг, екологічна ефективність.

Метою магістерської роботи є розробка додатку відеоспостереження, який дозволяє ефективно використовувати старі мобільні телефони для забезпечення безпеки та моніторингу. Основними завданнями є аналіз вимог та предметної області, проектування архітектури системи, вибір відповідних технологій та інструментів, реалізація програмного забезпечення для мобільного клієнта та клієнтської програми на ПК, а також проведення тестування для оцінки ефективності та надійності розробленого додатку.

Для досягнення поставлених завдань було обрано мови програмування Kotlin для розробки мобільного клієнта та Python для створення клієнтської програми на ПК. Архітектура системи базується на клієнт-серверній моделі, де мобільний клієнт захоплює відео та передає його на клієнтську програму через сервер сигналізації.

Проведене тестування показало, що розроблений додаток відповідає всім поставленим вимогам щодо продуктивності, стабільності та безпеки.

ABSTRACT

Development of a video broadcasting application for mobile devices based on the Android operating system // The educational level "Master" qualification work // Bondarenko Vladyslav Sergiyovych // Ternopil Ivan Pulyuy National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, SNm-61 group // Ternopil, 2024 // P. 78, fig. – 14, annexes – 2, ref. – 20.

Keywords: video surveillance, kotlin, python, kivy, webrtc, rtsp, client-server architecture, real-time, security, monitoring, environmental efficiency.

The purpose of the master's thesis is to develop a video surveillance application that allows the effective use of old mobile phones for security and monitoring. The main tasks are to analyze the requirements and subject area, design the system architecture, select appropriate technologies and tools, implement the software for the mobile client and the client program on the PC, and conduct testing to evaluate the effectiveness and reliability of the developed application.

To achieve the set tasks, we chose Kotlin programming language to develop a mobile client and Python to create a client program on a PC. The system architecture is based on a client-server model, where the mobile client captures video and transmits it to the client program through the alarm server.

The conducted testing has shown that the developed application meets all the requirements for performance, stability, and security.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД – база даних;

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер.

ЗМІСТ

ВСТУП

У сучасному суспільстві спостерігається постійне оновлення мобільних пристроїв, причому середній інтервал між їх замінами становить приблизно три роки. Цей динамічний розвиток мобільних технологій стимулює швидке впровадження нових функціональних можливостей та дизайну, що, з одного боку, забезпечує користувачам доступ до передових технологій, а з іншого — спричиняє значне накопичення не використовуваної техніки в домашніх умовах. Накопичення старих мобільних телефонів не лише займає фізичний простір, але й має негативний екологічний вплив через утилізацію електронних відходів, які містять шкідливі матеріали. Враховуючи ці виклики, виникає нагальна потреба у пошуку ефективних рішень для повторного використання електронних пристроїв, що сприятиме зменшенню екологічного навантаження та підвищенню економічної ефективності споживання ресурсів.

Одним із можливих шляхів вирішення цієї проблеми є розробка спеціалізованих застосунків, що сприяють повторному використанню мобільних телефонів, які більше не використовуються. У цьому контексті пропонується створення додатку відеоспостереження, який дозволяє перетворювати старі смартфони на ефективні інструменти для безпеки та моніторингу. Такий підхід не лише сприяє зменшенню кількості електронних відходів, але й додає цінності старим пристроям, роблячи їх корисними у повсякденному житті. Використання старих телефонів у цілях відеоспостереження відкриває широкі можливості для забезпечення додаткового рівня безпеки як у домашніх умовах, так і в автомобільній сфері, де такі пристрої можуть використовуватись як відеореєстратори для запису дорожніх подій.

Розробка додатку відеоспостереження включає в себе низку технічних та організаційних аспектів. З технічної точки зору, необхідно забезпечити стабільну передачу відео в реальному часі, зберігання записів та захист даних від несанкціонованого доступу. Це може бути досягнуто шляхом інтеграції сучасних технологій потокової передачі даних, хмарного зберігання та шифрування інформації. Організаційні аспекти включають розробку інтуїтивно зрозумілого користувацького інтерфейсу, який дозволить навіть некваліфікованим

користувачам легко налаштовувати та використовувати систему відеоспостереження.

Метою магістерської роботи є розробка додатку відеоспостереження, який дозволить ефективно використовувати старі мобільні телефони для забезпечення безпеки та моніторингу. Основними завданнями є:

- Дослідити предметну область та здійснити постановку проблеми;
- Розробити модель додатку відеоспостереження;
- Реалізувати програмне забезпечення для перетворення старих смартфонів у камери спостереження та відеореєстратори;
- Провести тестування розробленого додатку для оцінки його ефективності та надійності.

Для досягнення поставлених завдань планується провести аналіз існуючих рішень на ринку відеоспостереження, визначити їх переваги та недоліки, а також ідентифікувати ключові вимоги до нового застосунку. Наступним кроком є розробка технічної архітектури додатку, що включає вибір відповідних технологій та інструментів програмування. Реалізація програмного забезпечення здійснюватиметься з урахуванням принципів модульності та масштабованості, що дозволить легко інтегрувати додаткові функції у майбутньому. Тестування додатку проводитиметься на різних моделях смартфонів для забезпечення його сумісності та стабільної роботи у різних умовах експлуатації.

Тема даної магістерської роботи є надзвичайно актуальною, оскільки зростаюча кількість електронних відходів вимагає пошуку ефективних рішень для їх повторного використання та утилізації. Розробка додатку відеоспостереження сприятиме не лише екологічній відповідальності, але й підвищенню безпеки в суспільстві, забезпечуючи доступ до сучасних технологій без необхідності значних фінансових витрат. Впровадження такого рішення може суттєво зменшити екологічний слід та оптимізувати використання електронних ресурсів у сучасних домашніх умовах, одночасно надаючи користувачам додаткові можливості для моніторингу та захисту своєї власності.

В дипломній роботі буде використано мову програмування Kotlin та Python.

1 АНАЛІЗ ВИМОГ ТА ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка задачі

Завданням магістерської роботи є розробка додатку відеотрансляції з камери телефона на клієнт користувача на пк або іншому пристрої. Застосунок повинен підтримувати більшість андроїд пристроїв, а саме версії Android 8 та вище, що забезпечує більш широке використання.

Програмне забезпечення повинно врахувати апаратні можливості мобільних пристроїв аби мати можливість працювати з різноманітними відеокодеками. Крім того, необхідно забезпечити захищений зв'язок, щоб дані користувача не могли бути підмінені або перехоплені. Застосунок повинен бути енергоефективним для зменшення витрат батареї. Ключовою є можливість простого налаштування для передачі потоку відео на клієнт ПК.

Проаналізувавши предметну область, можемо виділити список задач, які необхідно вирішити:

- Розробка архітектури системи;
- Проєктування серверної частини;
- Пошук протоколу для передачі відеопотоку;
- Реалізація застосунку на android;
- Реалізація застосунку на ПК;
- Тестування системи.

Основні функції системи:

- Транслювання відеопотоку;
- Робота з розкадровкою стріму;
- Збільшення та зменшення гамми;
- Збільшення та зменшення зуму;
- Збільшення та зменшення контрасту;
- Збільшення та зменшення яскравості;
- Збереження кадрів;
- Масштабування по кадру.

Предметна область є актуальною з кількох ключових причин. По-перше, зростаюча інтеграція мобільних пристроїв у різні сфери діяльності, такі як освіта, медицина та безпека, вимагає ефективних рішень для передачі відео з телефонів на інші платформи. Забезпечення підтримки Android версії 8 та вище дозволяє охопити ширший спектр користувачів, включаючи власників старіших моделей пристроїв, що сприяє більшій доступності застосунку.

Крім того, врахування апаратних можливостей різних мобільних пристроїв забезпечує оптимальну продуктивність та стабільність відеотрансляції, незалежно від характеристик процесора, пам'яті та відеокодеків. Захищений зв’язок є критично важливим для запобігання підмінам та перехопленню даних, особливо при передачі конфіденційної інформації, що відповідає сучасним стандартам безпеки.

Енергоефективність застосунку також має велике значення, оскільки дозволяє зменшити витрати батареї мобільних пристроїв під час трансляції відео, що підвищує комфорт користування. Нарешті, інтуїтивно зрозумілий інтерфейс та простота налаштування сприяють широкому прийняттю застосунку серед користувачів з різними рівнями технічних знань.

Розробка додатку для відеотрансляції не лише відповідає сучасним вимогам ринку, а й активно задовольняє зростаючі потреби користувачів у мобільних рішеннях для обміну відеоконтентом. Висока доступність застосунку забезпечується підтримкою широкого спектра Android-пристроїв, включаючи старіші моделі, що розширює його потенційну аудиторію та сприяє масовому впровадженню. Це особливо важливо в умовах різноманітності ринку мобільних пристроїв, де користувачі використовують різні моделі з різними апаратними характеристиками.

Таким чином, розробка ПЗ для відеотрансляції відповідає сучасним вимогам ринку, забезпечуючи високу доступність, безпеку та ефективність, що підтверджує актуальність і перспективність даної предметної області.

1.2 Огляд вимог до програмного забезпечення

1.2.1 Функціональні вимоги

Функціональні вимоги визначають основні можливості та функції системи, що забезпечують її відповідність поставленим цілям та завданням. Для додатку відеотрансляції з камери мобільного пристрою на клієнтський додаток встановлено кілька ключових функціональних аспектів.

По-перше, система повинна забезпечувати безперервну передачу відеопотоку в реальному часі з високою якістю та мінімальною затримкою, що є критичним для забезпечення плавності перегляду. Крім того, необхідно надати можливість обробки відеопотоку на рівні окремих кадрів, що дозволяє здійснювати подальший аналіз або редагування зображення.

Користувачі повинні мати можливість регулювати параметри відео, такі як гамма, зум, контрастність та яскравість, через інтуїтивно зрозумілий інтерфейс у режимі реального часу. Це дозволяє адаптувати якість зображення до різних умов освітлення та вимог до деталізації. Також важливо забезпечити функціонал для збереження окремих кадрів відеопотоку у файлову систему клієнтського пристрою, а також адаптивне масштабування відео залежно від розміру екрана без втрати пропорцій.

1.2.2 Нефункціональні вимоги

Нефункціональні вимоги визначають характеристики системи, що впливають на якість її роботи та взаємодію з користувачем. Однією з ключових вимог є продуктивність, яка передбачає високу швидкість обробки та передачі відеопотоку з максимально допустимою затримкою не більше 200 мілісекунд при стандартних умовах мережі.

Сумісність системи з різними пристроями та операційними системами також є важливим аспектом, що забезпечує роботу додатку на більшості Android-пристроїв з версіями 8 та вище, а також на ПК та планшетах з різними апаратними характеристиками. Безпека передачі даних забезпечується

сучасними методами шифрування, такими як AES-256, та надійною автентифікацією користувачів для захисту від несанкціонованого доступу та перехоплення.

Енергоефективність додатку досягається оптимізацією використання ресурсів мобільного пристрою, що дозволяє зменшити витрати батареї. Масштабованість системи забезпечується модульною архітектурою, що дозволяє легко розширювати функціонал та підтримувати більшу кількість користувачів без суттєвих змін у структурі системи. Надійність системи гарантується стабільною роботою додатку без збоїв та втрат даних завдяки механізмам відновлення після помилок та резервного копіювання.

Зручність використання (юзабіліті) забезпечується інтуїтивно зрозумілим інтерфейсом, який дозволяє користувачам легко налаштовувати та використовувати додаток без необхідності глибоких технічних знань. Нарешті, підтримка та обслуговування системи передбачають можливість легкого оновлення додатку та виправлення помилок за допомогою стандартних методів оновлення та ведення детальної документації.

1.3 Розгляд аналогів

1.3.1 Програма Camy

Camy – один з додатків для перетворення телефону на камеру спостереження (рис. 1.1). Camy є конкурентним продуктом у сфері систем відеоспостереження, який перетворює мобільні телефони та планшети у системи відеоспостереження в реальному часі. Ця програма дозволяє користувачам здійснювати підключення до іншого мобільного пристрою з будь-якої локації за умови наявності інтернет-з'єднання, що забезпечує високу ступінь мобільності та доступності.

Camy надає можливість моніторингу житлових приміщень або офісних просторів без необхідності придбання спеціалізованого обладнання. Це економічно ефективне рішення, яке використовує наявні мобільні пристрої для створення системи відеоспостереження. Завдяки цьому користувачі можуть

здійснювати спостереження за дітьми, домашніми тваринами або перевіряти, хто знаходиться вдома в поточний момент часу.

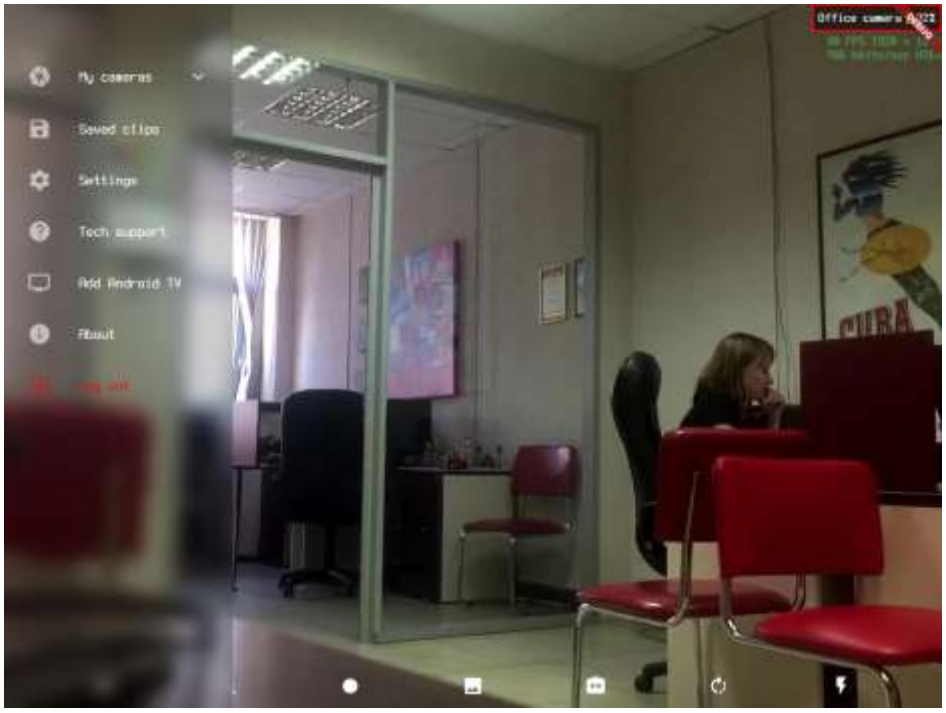

Рисунок 1.1 – Вікно застосунку Camy

Однією з ключових функцій Camy є вбудований детектор руху, який підвищує рівень безпеки, автоматично виявляючи несанкціоновані проникнення та негайно повідомляючи користувача про такі події. Це забезпечує оперативну реакцію на потенційні загрози, що є особливо важливим для забезпечення безпеки житлових приміщень та комерційних об'єктів.

Camy підтримує передачу відеопотоку в реальному часі з високою якістю зображення та мінімальною затримкою, що забезпечує стабільний та надійний моніторинг. Програма також забезпечує можливість збереження відеозаписів, що дозволяє користувачам здійснювати подальший аналіз подій або підтвердження інцидентів.

Інтерфейс програми Camy є інтуїтивно зрозумілим, що дозволяє користувачам з мінімальними технічними знаннями легко налаштовувати та використовувати додаток. Простота налаштування та використання сприяє широкому прийняттю програми серед різних груп користувачів, що є важливим фактором для її успішної інтеграції на ринку.

Таким чином, Camy демонструє ефективне використання мобільних технологій для створення доступних та функціональних систем відеоспостереження. Завдяки інтеграції функцій реального часу та автоматичного виявлення руху, Camy забезпечує надійний моніторинг без необхідності значних фінансових витрат на інфраструктуру відеоспостереження. Це робить Camy привабливим рішенням для користувачів, які шукають гнучкі та економічно вигідні способи забезпечення безпеки своїх приміщень.

1.3.2 Програма Alfred Camera

Alfred Camera є одним із провідних аналогів Camy у сфері мобільних систем відеоспостереження (рис. 1.2).

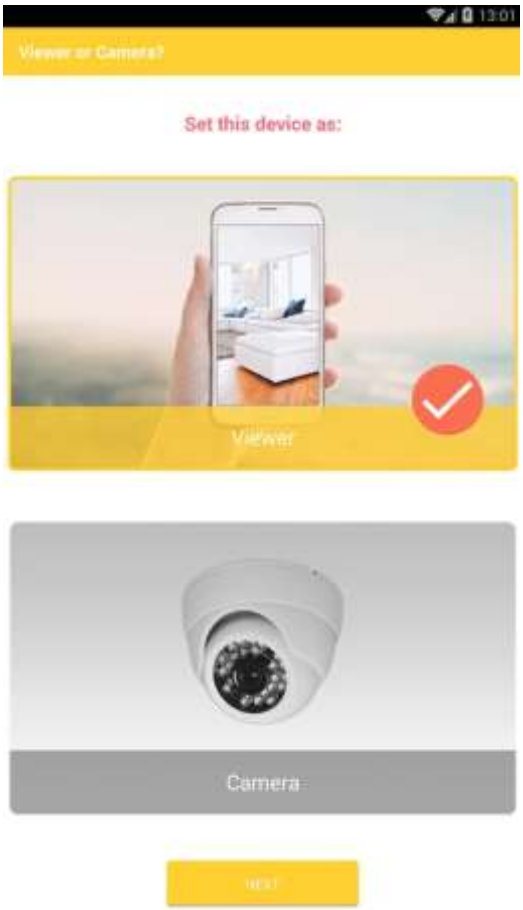

Рисунок 1.2 – Вікно застосунку Alfred Camera при виборі реєстрації пристрою

Цей додаток дозволяє перетворити старі смартфони та планшети у бездротові камери спостереження, забезпечуючи трансляцію відео в реальному часі на інші пристрої. Основними функціями Alfred Camera є

відеоспостереження в реальному часі, яке забезпечує високу якість відеопотоку з можливістю перегляду через мобільний додаток або веб-браузер, детектор руху та звуку, який автоматично виявляє підозрілу активність та надсилає сповіщення користувачу, нічний режим, що використовує інфрачервоне освітлення для спостереження в умовах низької освітленості, а також двосторонній аудіо, який дозволяє користувачам не лише спостерігати, але й спілкуватися через вбудований мікрофон та динамік.

Перевагами Alfred Camera є простота налаштування та використання, наявність безкоштовної базової версії з можливістю розширення функціоналу через преміум-підписку, а також підтримка хмарного зберігання відеозаписів. Однак серед недоліків слід відзначити обмежені можливості кастомізації у безкоштовній версії та залежність від стабільного інтернет-з'єднання для оптимальної роботи. Alfred Camera користується популярністю серед користувачів завдяки своїй доступності та широкому набору функцій, що робить його конкурентоспроможним продуктом на ринку мобільних систем відеоспостереження.

1.3.3 Програма Manything

Manything, абревіатура від "Monitoring Anything", є ще одним застосунком який пропонує розширені можливості відеоспостереження через мобільні пристрої (рис. 1.3). Основними функціями Manything є відеозапис та зберігання, що дозволяє записувати відео з мобільних пристроїв та зберігати їх у хмарі для подальшого перегляду, інтеграція з іншими пристроями, що дозволяє підключати до нього інші камери та інтегруватися з системами розумного дому, детектор руху та подій, який автоматично виявляє рух з можливістю налаштування чутливості та зон спостереження, а також система повідомлень у режимі реального часу, яка надсилає сповіщення на мобільний пристрій при виявленні підозрілих подій.

Рисунок 1.3 – Головне вікно програми Manything

Перевагами Manything є широкі можливості кастомізації та налаштування сповіщень, підтримка декількох користувачів для спільного доступу до камер, а також інтеграція з популярними платформами розумного дому, такими як Amazon Alexa та Google Assistant. Серед недоліків можна відзначити вартість преміум-підписки для доступу до розширених функцій та високі вимоги до зберігання даних у хмарі при великій кількості відеозаписів. Manything вирізняється своєю гнучкістю та можливістю інтеграції з іншими системами, що робить його привабливим вибором для користувачів, які шукають більш комплексні рішення для відеоспостереження.

1.4 Вибір моделі розробки

Інкрементальна модель розробки програмного забезпечення передбачає поступове створення системи шляхом додавання нових функціональних компонентів до вже існуючої базової версії продукту. Кожен інкремент включає розробку певного набору функцій, які інтегруються з попередніми версіями, забезпечуючи поступове вдосконалення системи. Цей підхід дозволяє розробникам створювати робочі частини системи на ранніх етапах проекту, що сприяє більш ефективному управлінню ресурсами та адаптації до змін вимог.

Додатково, інкрементальна модель сприяє зниженню ризиків, пов'язаних із розробкою великих та складних програмних систем. Оскільки кожен інкремент розробляється та тестується окремо, це дозволяє виявляти та виправляти дефекти на ранніх стадіях, зменшуючи витрати на їх усунення та підвищуючи загальну якість кінцевого продукту. Такий підхід також забезпечує можливість проведення ретельного аналізу кожного інкременту перед його інтеграцією до основної системи, що сприяє більш стабільній та надійній роботі програмного забезпечення.[14]

Основні етапи моделі розробки поділяються на визначення вимог для інкременту, проектування та розробку, інтеграцію та тестування, а також реліз інкременту.

Визначення вимог для інкременту: На цьому етапі проводиться аналіз та уточнення вимог, які повинні бути реалізовані в поточному інкременті. Визначаються пріоритети функціоналу та його взаємодія з існуючою системою.

Проектування та розробка: Розробники проектують архітектуру нових функцій та здійснюють їх реалізацію відповідно до визначених вимог. Цей етап включає написання коду, розробку інтерфейсів та інтеграцію нових компонентів з вже існуючими частинами системи.

Інтеграція та тестування: Після розробки нових функцій вони інтегруються з попередніми інкрементами. Проводиться комплексне тестування для забезпечення сумісності та стабільності системи, а також для виявлення та виправлення можливих помилок.

Реліз інкременту: Завершальний етап включає розгортання нового інкременту на робочі середовища користувачів. Після релізу здійснюється збір зворотного зв'язку для подальшого вдосконалення системи в наступних інкрементах.

Переваги:

- Гнучкість у зміні вимог та додаванні нових функцій: Інкрементальний підхід дозволяє легко адаптуватися до змін вимог протягом всього процесу розробки, що є особливо корисним у динамічних проектах.
- Швидка доставка робочих частин системи: Завдяки поступовій реалізації функціоналу користувачі отримують доступ до робочих частин

системи на ранніх етапах, що сприяє швидкому отриманню цінності від продукту.

- Легше виявлення та виправлення помилок на ранніх етапах: Поступова інтеграція та тестування нових функцій дозволяє виявляти та виправляти помилки на ранніх стадіях, що зменшує ризик серйозних проблем у майбутньому.

Недоліки:

- Можлива нестабільність системи через часті інтеграції: Постійна інтеграція нових функцій може призводити до тимчасової нестабільності системи, особливо якщо інтеграції виконуються без належного тестування.
- Потребує ретельного планування для забезпечення узгодженості між інкрементами: Для успішної реалізації інкрементальної моделі необхідно ретельно планувати кожен інкремент, щоб забезпечити сумісність нових функцій з уже існуючими компонентами системи.
- Може виникнути складність при управлінні великою кількістю інкрементів: У великих проектах, де реалізовується багато інкрементів, може виникати складність у їхньому управлінні, що потребує ефективних інструментів та методів контролю версій.

Інкрементальна модель розробки програмного забезпечення є ефективним підходом для проектів, де вимоги можуть змінюватися протягом часу, а також для проектів, що потребують швидкої доставки робочих частин системи. Ця модель дозволяє гнучко реагувати на зміни, забезпечує поступову інтеграцію функціоналу та сприяє ранньому виявленню помилок. Проте, для успішної реалізації інкрементальної моделі необхідно ретельно планувати кожен інкремент та забезпечити ефективне управління процесом розробки, щоб уникнути можливих проблем із сумісністю та стабільністю системи.

Перейдемо до обгрунтування вибору технологій та інструментів розробки проєкту.

1.5 Вибір технологій та інструментів

1.5.1 Мова програмування Kotlin

Для написання магістерської роботи було обрано мову програмування Kotlin. Ця мова ідеально підходить для розробки сучасних мобільних та серверних додатків, а також має всі необхідні та зручні робочі інструменти для ефективної розробки програмного забезпечення.

Kotlin є сучасною, статично типізованою мовою програмування, яка поєднує в собі функціональні та об'єктно-орієнтовані парадигми. Вона була розроблена компанією JetBrains та вперше представлена у 2011 році. Kotlin офіційно підтримується Google для розробки Android-додатків, що зробило її однією з найпопулярніших мов для мобільної розробки.

Kotlin є відкритим кодом, що робить її доступною для широкої спільноти розробників. Мова відзначається простотою, сучасністю, гнучкістю та універсальністю, що дозволяє використовувати її для створення різноманітного програмного забезпечення. Kotlin сумісна з JVM (Java Virtual Machine), що дозволяє інтегруватися з існуючим Java-кодом та використовувати всі переваги екосистеми Java.

Мова програмування – це інструмент, який використовується для написання програмних програм. Kotlin забезпечує ефективну розробку завдяки своєму лаконічному синтаксису, який дозволяє зменшити кількість шаблонного коду порівняно з Java. Це сприяє підвищенню продуктивності розробників та зменшенню ймовірності виникнення помилок.

Ключові характеристики мови Kotlin включають:

- Сучасний та лаконічний синтаксис: Kotlin дозволяє писати менш об'ємний та більш читабельний код, що спрощує процес розробки та підтримки програмного забезпечення.
- Безпека типів: Система типів Kotlin допомагає уникнути поширених помилок, таких як NullPointerException, завдяки вбудованій підтримці нульових типів.

- Інтероперабельність з Java: Kotlin повністю сумісна з Java, що дозволяє легко інтегруватися з існуючим Java-кодом та використовувати всі бібліотеки та фреймворки Java.
- Підтримка функціонального програмування: Мова підтримує лямбда-вирази, функції вищого порядку та інші функціональні концепції, що дозволяє писати більш чистий та ефективний код.
- Корутини: Вбудована підтримка корутин дозволяє ефективно управляти асинхронними операціями та покращує продуктивність додатків.
- Мультиплатформені можливості: Kotlin підтримує розробку мультиплатформних додатків, що дозволяє використовувати один кодовий базис для різних платформ, таких як Android, iOS, веб та серверні додатки.

Мова Kotlin була створена з метою задовольнити потреби бізнесу та підприємств, забезпечуючи розробників гнучкими та сучасними інструментами для створення високоякісного програмного забезпечення. Вона підходить для розробки всіх видів програмного забезпечення, включаючи клієнтські додатки, серверні компоненти, бібліотеки, сервіси та API, веб-додатки, мобільні додатки, хмарні програми та відеоігри.

Ключові моменти, які допомагають писати безпечний та ефективний код у Kotlin, включають:

- Заборона на небезпечні перетворення типів: Kotlin не дозволяє перетворення типів, які можуть призвести до втрати даних або інших проблем, що сприяє написанню безпечного коду.
- Підтримка типів Null: Мова забезпечує чітке розмежування між типами, що можуть містити null, та типами, що не можуть, що допомагає уникнути помилок, пов'язаних із нульовими значеннями.
- Повернення з типом «тільки для читання»: Це дозволяє використовувати копії об'єктів під час компіляції, підвищуючи безпеку та ефективність коду.
- Оптимізація передачі структур: Kotlin рекомендує передавати структури як параметри з міркувань продуктивності та уникати передачі великих структур без необхідності.

Також важливою є можливість програмування між платформами. Kotlin дозволяє створювати програми, які можна розгортати на різних операційних системах, таких як Windows, Linux та macOS, а також у хмарі та контейнерах. Це робить Kotlin універсальною мовою програмування, яка підходить для різних видів проектів та платформ.

Крім того, Kotlin постійно розвивається, отримуючи нові можливості та покращення, що дозволяє розробникам створювати сучасні та ефективні програмні системи, які будуть актуальними та застосовуваними у майбутньому [1].

1.5.2 Мова програмування Python

Для написання клієнту на ПК було обрано мову програмування Python. Ця мова ідеально підходить для розробки різноманітних додатків, включаючи веб-сайти, наукові обчислення, автоматизацію, аналіз даних та штучний інтелект. Python має потужний набір бібліотек та зручні робочі інструменти, що сприяють ефективній розробці програмного забезпечення.

Python є високорівневою, інтерпретованою мовою програмування з динамічною типізацією, яка підтримує кілька парадигм програмування, включаючи об'єктно-орієнтоване, процедурне та функціональне програмування. Вона була створена Гвідо ван Россумом і вперше представлена у 1991 році. Python здобув широку популярність завдяки своїй простоті, читабельності коду та великій спільноті розробників.

Python є відкритим програмним забезпеченням, що робить його доступним для широкої спільноти розробників по всьому світу. Мова відзначається простотою синтаксису, що дозволяє швидко писати та читати код, а також високою гнучкістю, що дозволяє використовувати її в різних сферах програмування. Python сумісний з різними операційними системами, такими як Windows, Linux та macOS, що забезпечує універсальність його застосування.[2]

Мова програмування – це інструмент, який використовується для написання програмних програм. Python забезпечує ефективну розробку завдяки своєму простому та зрозумілому синтаксису, який дозволяє зменшити кількість

коду порівняно з іншими мовами програмування. Це сприяє підвищенню продуктивності розробників та зменшенню ймовірності виникнення помилок.

Ключові характеристики мови Python включають:

- Простий та читабельний синтаксис: Python дозволяє писати зрозумілий та легко підтримуваний код, що спрощує процес розробки та співпраці в командах.
- Велика стандартна бібліотека: Мова постачається з широким набором вбудованих модулів, що дозволяє швидко реалізовувати різноманітні функціональні можливості без необхідності написання додаткового коду.
- Інтерпретована мова: Python виконується рядок за рядком, що дозволяє швидко тестувати та налагоджувати код, а також спрощує процес розробки.
- Підтримка різних парадигм програмування: Мова підтримує об'єктно-орієнтоване, процедурне та функціональне програмування, що дозволяє розробникам обирати найбільш підходящий підхід для вирішення конкретних завдань.
- Платформонезалежність: Python може виконуватися на різних операційних системах без необхідності зміни коду, що забезпечує його універсальність.
- Розширюваність: Можливість інтеграції з іншими мовами програмування, такими як C та C++, дозволяє розширювати функціональні можливості Python та оптимізувати продуктивність.
- Велика спільнота та підтримка: Активна спільнота розробників забезпечує постійне оновлення мови, створення нових бібліотек та фреймворків, а також надання підтримки новачкам та професіоналам.

Мова Python була створена з метою забезпечити простоту та ефективність програмування, задовольняючи потреби різних сфер бізнесу та наукових досліджень. Вона підходить для розробки всіх видів програмного забезпечення, включаючи веб-додатки, наукові обчислення, машинне навчання, автоматизацію процесів, обробку даних, створення скриптів та багато іншого.

Ключові моменти, які допомагають писати безпечний та ефективний код у Python, включають:

- Динамічна типізація: Хоча це забезпечує гнучкість, Python також підтримує використання статичної типізації через додаткові інструменти, такі як type hints, що сприяє підвищенню безпеки коду.
- Управління пам'яттю: Автоматичне управління пам'яттю через збірку сміття дозволяє уникати проблем, пов’язаних з ручним управлінням пам’яттю.
- Виключення та обробка помилок: Механізм обробки виключень дозволяє ефективно управляти помилками та забезпечувати стабільність роботи програм.
- Підтримка тестування: Вбудовані інструменти та бібліотеки для тестування сприяють написанню надійного та перевіреного коду.
- Документування коду: Можливість створення докстрінгів та використання інструментів для автоматичного створення документації допомагає підтримувати якість та зрозумілість коду.

Також важливою є можливість програмування між платформами. Python дозволяє створювати програми, які можна розгортати на різних операційних системах, а також використовувати в хмарних середовищах та контейнерах. Це робить Python універсальною мовою програмування, яка підходить для різних видів проектів та платформ.

Крім того, Python постійно розвивається, отримуючи нові можливості та покращення, що дозволяє розробникам створювати сучасні та ефективні програмні системи, які будуть актуальними та застосовуваними у майбутньому. Активний розвиток мови та її екосистеми забезпечує її тривалу популярність та застосовність у різних галузях технологій.

1.5.3 Фреймворк Kivy

Цей фреймворк є потужним інструментом для розробки кросплатформених додатків з підтримкою мультитач, що робить його ідеальним вибором для створення сучасних мобільних та десктопних програм. Kivy надає розробникам усі необхідні компоненти та зручні робочі інструменти для ефективної та швидкої розробки програмного забезпечення [3].

Kivy є відкритим фреймворком з використанням мови програмування Python, що дозволяє легко інтегрувати його з іншими бібліотеками та інструментами екосистеми Python. Він був розроблений з метою забезпечити високопродуктивну та гнучку платформу для створення інтерфейсів користувача, що підтримують сучасні технології взаємодії, такі як жестове управління та сенсорні екрани.

Kivy є кросплатформним, що означає, що додатки, створені за його допомогою, можуть бути запущені на різних операційних системах, таких як Windows, macOS, Linux, Android та iOS, без необхідності внесення значних змін у код. Це забезпечує універсальність та широку застосовність розроблених додатків.

Фреймворк Kivy відзначається простотою використання та високою гнучкістю, що дозволяє розробникам швидко прототипувати та реалізовувати складні функціональні можливості. Завдяки архітектурі, що підтримує MVC (Model-View-Controller) та MVVM (Model-View-ViewModel) патерни, Kivy сприяє організованій та підтримуваній розробці програмного забезпечення.

Мова програмування Python, на якій базується Kivy, забезпечує простий та читабельний синтаксис, що дозволяє зменшити кількість коду та підвищити продуктивність розробників. Це особливо важливо при розробці складних додатків, де чистота та зрозумілість коду мають критичне значення для підтримки та масштабування проекту.

Ключові характеристики фреймворку Kivy включають:

- Кросплатформені можливості: Підтримка різних операційних систем, включаючи мобільні платформи, що дозволяє розробляти універсальні додатки з одним кодовим базисом.
- Підтримка мультитач: Вбудована підтримка сенсорних екранів та жестів, що дозволяє створювати інтуїтивно зрозумілі інтерфейси користувача.
- Висока продуктивність: Використання OpenGL ES2 для рендерингу графіки забезпечує плавну та швидку роботу додатків.
- Гнучкий інтерфейс користувача: Можливість створення складних та адаптивних інтерфейсів за допомогою декларативного мови розмітки Kivy Language (KV).

- Багатий набір віджетів: Велика бібліотека вбудованих віджетів, таких як кнопки, текстові поля, списки та інші елементи інтерфейсу, що спрощують процес розробки.
- Модульність та розширюваність: Легкість інтеграції з іншими бібліотеками та можливість створення власних віджетів та модулів.
- Активна спільнота та підтримка: Велика та активна спільнота розробників, яка постійно розвиває фреймворк, створює нові бібліотеки та надає підтримку через форуми та документацію.

Фреймворк Kivy був створений з метою задовольнити потреби сучасних розробників у створенні високоякісних та ефективних додатків. Він підходить для розробки різноманітних видів програмного забезпечення, включаючи мобільні додатки, ігри, навчальні програми, бізнес-додатки та інші інтерактивні системи.

Ключові моменти, які допомагають писати безпечний та ефективний код у Kivy, включають:

- Архітектурна підтримка MVC/MVVM: Забезпечує чітке розділення логіки програми та інтерфейсу користувача, що сприяє написанню чистого та підтримуваного коду.
- Управління ресурсами: Ефективне управління пам’яттю та ресурсами додатка завдяки оптимізації використання графічних ресурсів та інших компонентів системи.
- Обробка подій та асинхронність: Вбудовані механізми обробки подій та підтримка асинхронних операцій, що дозволяє створювати плавні та відзивчиві інтерфейси.
- Документування та тестування: Можливість створення детальної документації для коду та використання інструментів для автоматизованого тестування, що підвищує якість та надійність додатків.
- Безпека типів: Використання Python з підтримкою типізації через type hints, що допомагає уникати помилок та підвищує безпеку коду.

Також важливою є можливість програмування між платформами. Kivy дозволяє створювати програми, які можна розгортати на різних операційних системах, мобільних пристроях, а також у хмарних середовищах та контейнерах.

Це робить Kivy універсальним фреймворком, який підходить для різних видів проектів та платформ.

Крім того, фреймворк Kivy постійно розвивається, отримуючи нові можливості та покращення, що дозволяє розробникам створювати сучасні та ефективні програмні системи, які залишаються актуальними та застосовними у майбутньому. Активний розвиток фреймворку та його екосистеми забезпечує його тривалу популярність та широке застосування у різних галузях технологій.

Таким чином, фреймворк Kivy є потужним та гнучким інструментом для розробки сучасних кросплатформених додатків, що забезпечує високу продуктивність, зручність використання та підтримку сучасних технологій взаємодії, роблячи його ідеальним вибором для магістерської роботи та професійної розробки програмного забезпечення.

1.5.4 WebRTC

WebRTC (Web Real-Time Communication) є сучасним протоколом, розробленим для забезпечення безпосередньої передачі аудіо, відео та даних у реальному часі між веб-браузерами та мобільними додатками без необхідності використання додаткових плагінів чи стороннього програмного забезпечення. Основною метою WebRTC є спрощення процесу створення інтерактивних комунікаційних сервісів, таких як відеоконференції, голосові дзвінки та обмін файлами, забезпечуючи при цьому високу якість передачі та низьку затримку [4].

Архітектура WebRTC складається з кількох ключових компонентів, включаючи API для доступу до медіа-потоків, механізми встановлення з'єднання та протоколи для передачі даних. Основні технології, що лежать в основі WebRTC, включають STUN (Session Traversal Utilities for NAT) та TURN (Traversal Using Relays around NAT) сервери, які дозволяють ефективно обходити мережеві обмеження та встановлювати з'єднання між користувачами, навіть у складних мережевих умовах.

Безпека є однією з пріоритетних характеристик WebRTC. Всі передані дані зашифровані за допомогою протоколів DTLS (Datagram Transport Layer Security) та SRTP (Secure Real-Time Transport Protocol), що забезпечує захист інформації

від несанкціонованого доступу та перехоплення. Крім того, WebRTC підтримує автентифікацію користувачів та контроль доступу до медіа-потоків, що додатково підвищує рівень безпеки комунікацій.

WebRTC також підтримує різні кодеки для аудіо та відео, що дозволяє адаптуватися до різних умов мережі та забезпечувати оптимальну якість передачі. Завдяки механізмам адаптивного бітрейту та зміни параметрів потоку в реальному часі, WebRTC може динамічно регулювати якість медіа-потоків, забезпечуючи стабільну роботу навіть при змінних мережевих умовах.

Завдяки своїй відкритій архітектурі та активній підтримці спільноти розробників, WebRTC постійно розвивається, впроваджуючи нові функції та покращення. Це дозволяє протоколу залишатися актуальним та ефективним інструментом для створення сучасних комунікаційних систем, що відповідають вимогам швидко зростаючого ринку та потреб користувачів.

1.5.5 Протокол RTSP

RTSP (Real Time Streaming Protocol) є протоколом, розробленим для управління потоковим мультимедійним контентом, таким як відео та аудіо, з можливістю контролю над потоками, включаючи функції відтворення, паузи та зупинки. Основною метою RTSP є забезпечення ефективного керування потоковими даними між клієнтом та сервером, що робить його популярним вибором для систем відеоспостереження, медіасерверів та інших додатків, де необхідно передавати великий обсяг медіа-даних.

Проте, при порівнянні RTSP з WebRTC, стає очевидним, що WebRTC пропонує ряд переваг, особливо в контексті сучасних комунікаційних потреб. По-перше, WebRTC забезпечує низьку затримку передачі даних, що є критично важливим для інтерактивних додатків, таких як відеоконференції та онлайн-ігри. RTSP, з іншого боку, більше орієнтований на передачу медіа-контенту з дещо вищою затримкою, що може бути менш придатним для реального часу взаємодії.

По-друге, WebRTC підтримує двосторонню комунікацію без необхідності встановлення додаткових з'єднань або використання проміжних серверів, що спрощує архітектуру системи та зменшує навантаження на сервери. RTSP

зазвичай потребує окремих серверів для керування потоками та передаванням даних, що може ускладнювати розгортання та масштабування систем.

Крім того, WebRTC забезпечує вбудовану підтримку сучасних стандартів безпеки, таких як шифрування даних за допомогою DTLS та SRTP, що гарантує захист інформації під час передачі. RTSP також може використовувати шифрування, проте його впровадження менш уніфіковане та може вимагати додаткових налаштувань для забезпечення повної безпеки.

Ще однією важливою перевагою WebRTC є його сумісність з сучасними веб-технологіями та легка інтеграція з різними платформами, включаючи мобільні пристрої та веб-браузери. Це дозволяє розробникам створювати універсальні додатки, які можуть працювати на різних пристроях без необхідності адаптації під конкретні протоколи передачі даних. RTSP, навпаки, часто обмежений певними платформами та потребує додаткових клієнтських програм для повноцінної роботи.

Враховуючи всі ці аспекти, WebRTC стає кращим вибором для розробки сучасних додатків, що потребують інтерактивної та безпечної передачі даних у реальному часі. Його гнучкість, ефективність та підтримка сучасних стандартів роблять WebRTC більш придатним для реалізації високоякісних комунікаційних систем порівняно з RTSP.

1.6 Середовище розробки Android Studio

Android Studio є офіційним інтегрованим середовищем розробки (IDE) для створення додатків під платформу Android, розробленим компанією Google. Базуючись на потужній платформі IntelliJ IDEA від JetBrains, Android Studio надає розробникам широкий спектр інструментів для ефективного створення, налагодження та оптимізації мобільних додатків. Основними мовами програмування, що підтримуються в Android Studio, є Java та Kotlin, що дозволяє розробникам вибирати найбільш зручну та сучасну мову для реалізації функціональних можливостей додатків.

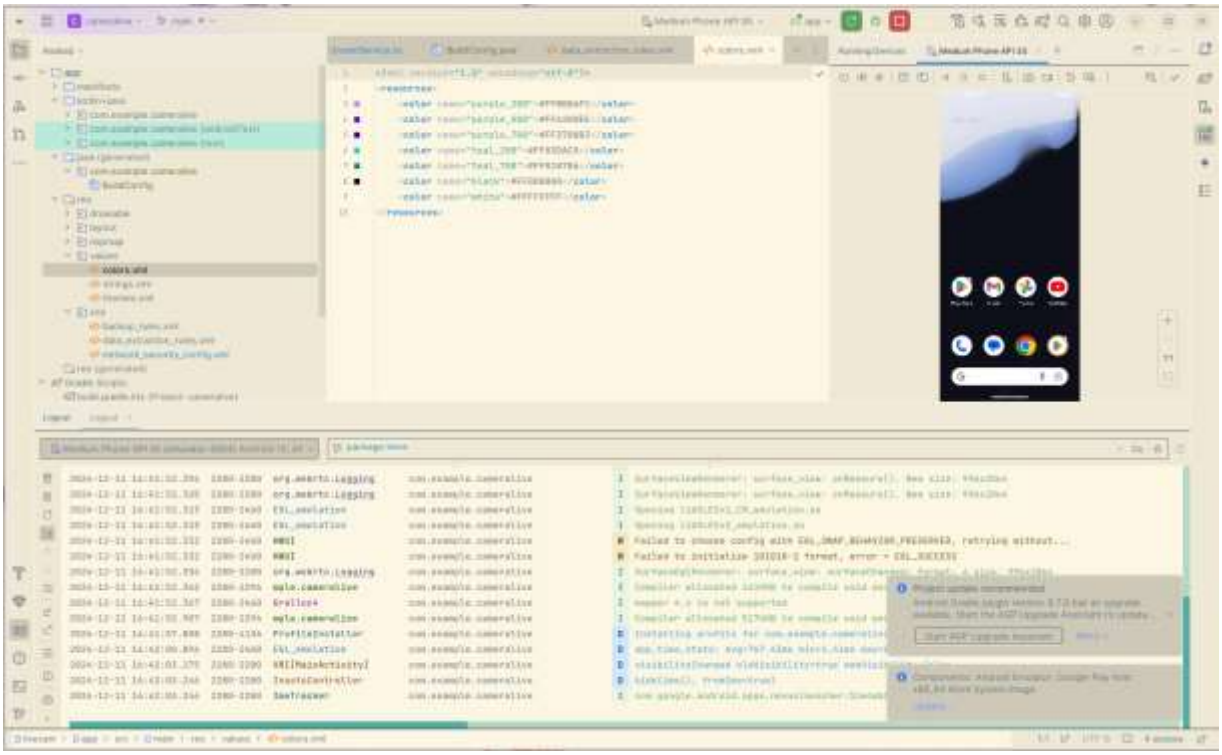

Рисунок 1.4 - Головне вікно Android Studio

Однією з ключових особливостей Android Studio є інтеграція з системою автоматизованої збірки Gradle, яка спрощує керування залежностями проекту та налаштування процесу збірки. Gradle дозволяє автоматизувати різні етапи розробки, включаючи компіляцію, тестування та розгортання додатків, що значно підвищує продуктивність розробників та забезпечує гнучкість у конфігуруванні проектів для різних середовищ та варіантів використання.

Вбудовані емулятори Android Studio надають можливість тестування додатків на різних пристроях та версіях операційної системи Android без необхідності фізичного обладнання. Це забезпечує розробникам змогу швидко перевіряти функціональність додатків у різних умовах, що сприяє виявленню та усуненню помилок на ранніх етапах розробки. Крім того, Android Studio підтримує інтеграцію з популярними системами контролю версій, такими як Git, що полегшує співпрацю в команді та управління змінами в коді проекту.

Редактор коду в Android Studio пропонує інтуїтивно зрозумілий інтерфейс з підсвічуванням синтаксису, автозаповненням та можливостями рефакторингу, що сприяє написанню чистого та оптимізованого коду. Інструменти для налагодження, включаючи дебаггер та профайлер, дозволяють розробникам детально аналізувати продуктивність додатків, виявляти та виправляти помилки, а також оптимізувати використання ресурсів пристрою.[6]

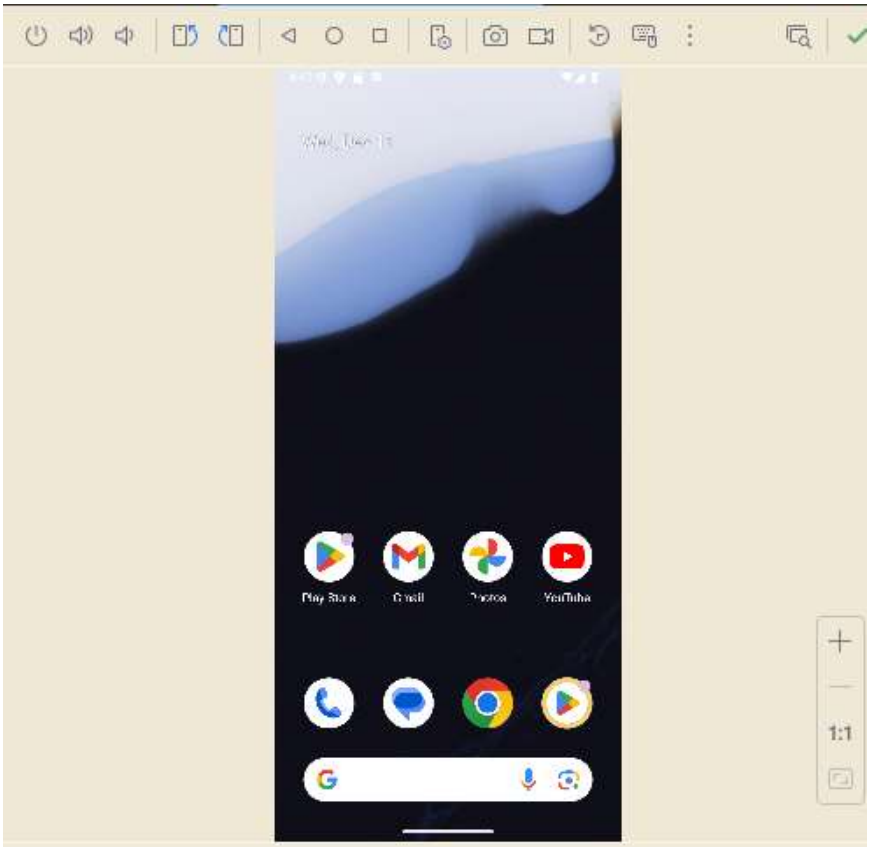

Рисунок 1.5 - Вікно емулятора Android пристрою

Особливою перевагою Android Studio є дизайнер інтерфейсу користувача, який дозволяє створювати та редагувати макети додатків за допомогою візуального редактора з функцією перетягування елементів. Це значно спрощує процес дизайну, дозволяючи розробникам швидко створювати адаптивні та привабливі інтерфейси, що відповідають сучасним стандартам користувацького досвіду.

У контексті розробки системи відеотрансляції з використанням WebRTC, Android Studio забезпечує необхідні інструменти для інтеграції медіа-даних, оптимізації мережевих з'єднань та забезпечення стабільної роботи додатку на різних пристроях. Завдяки підтримці WebRTC API та можливості використання нативних бібліотек, розробники можуть реалізовувати ефективні алгоритми передачі та обробки відео- та аудіопотоків, що забезпечує високу якість комунікації та мінімальні затримки.

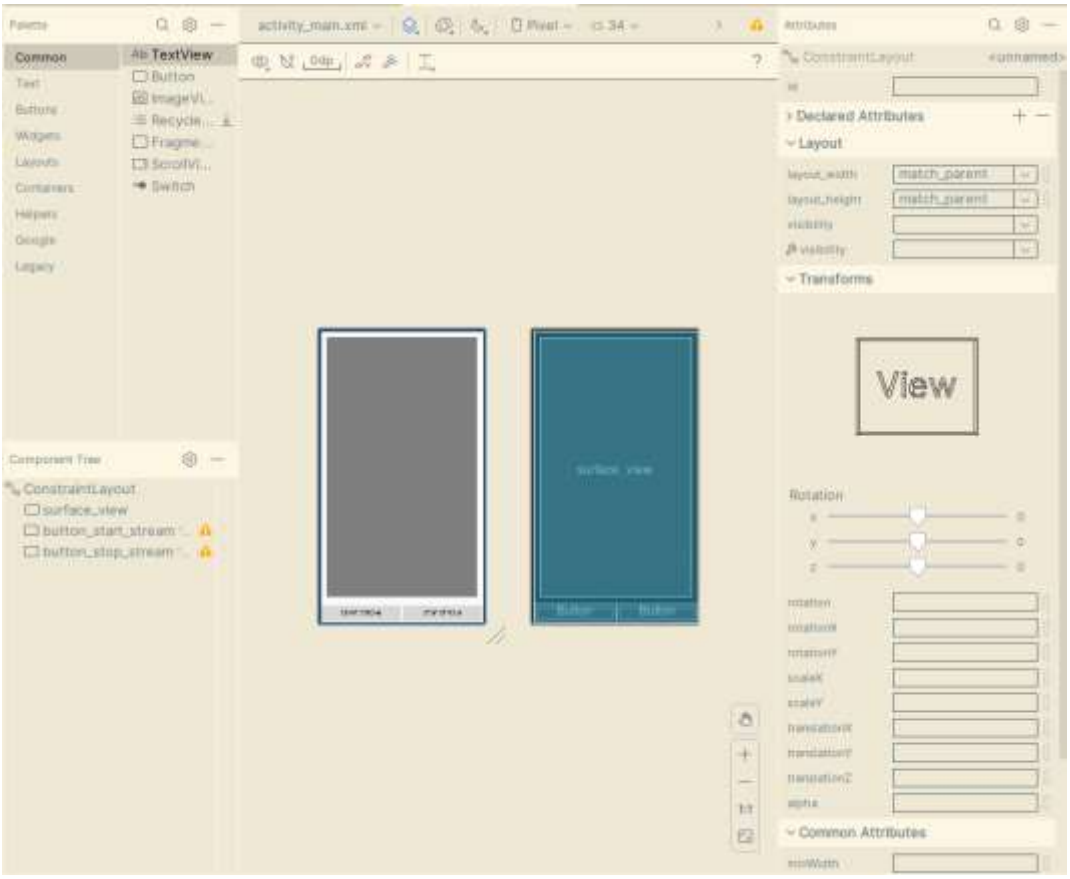

Рисунок 1.6 - Вікно дизайнер інтерфейсу

1.7 Середовище розробки PyCharm

Середовище розробки PyCharm є одним із провідних інтегрованих середовищ розробки (IDE) для мови програмування Python, розробленим компанією JetBrains. Воно призначене для забезпечення розробників потужними інструментами, які спрощують процес створення, налагодження та оптимізації Python-додатків. PyCharm підтримує як стандартні бібліотеки Python, так і численні фреймворки, що робить його універсальним інструментом для розробки в різних сферах, включаючи веб-розробку, наукові дослідження, автоматизацію та аналіз даних.

Однією з ключових особливостей PyCharm є його інтелектуальний редактор коду, який пропонує підсвічування синтаксису, автозаповнення коду та можливості рефакторингу. Це дозволяє розробникам писати чистий та ефективний код швидше та з меншими помилками. Крім того, PyCharm надає потужні інструменти для налагодження, включаючи інтегрований дебаггер та профайлер, що дозволяють детально аналізувати виконання програм, виявляти та усувати помилки, а також оптимізувати продуктивність додатків.

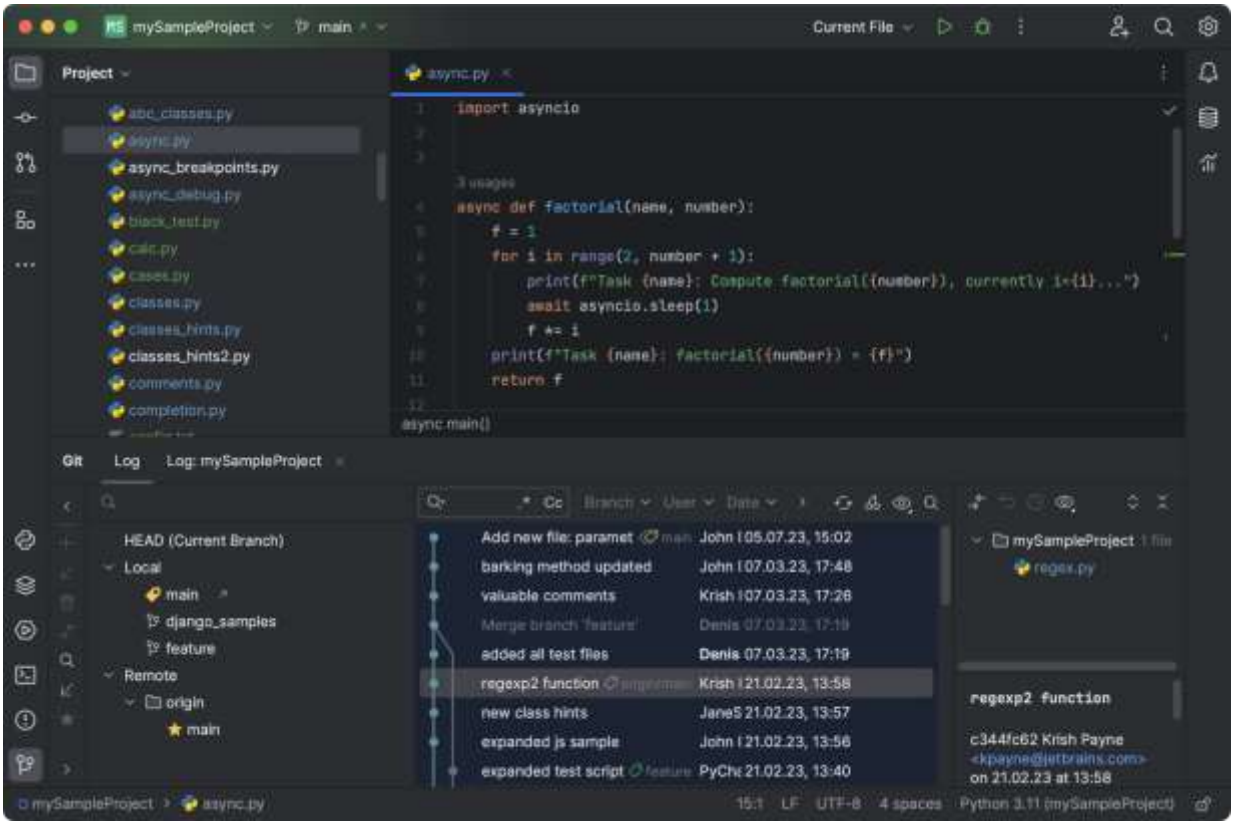

Рисунок 1.7 - Головне вікно програми Pycharm

Інтеграція з системами контролю версій, такими як Git, спрощує співпрацю в команді та управління змінами в коді проекту. PyCharm підтримує роботу з віртуальними середовищами, такими як virtualenv та Conda, що дозволяє ізолювати залежності проектів та забезпечує чистоту середовища розробки. Інтеграція з пакетними менеджерами, такими як pip, спрощує встановлення та управління бібліотеками та фреймворками, необхідними для проектів.

PyCharm пропонує вбудовану підтримку численних популярних фреймворків та технологій, таких як Django, Flask та Pyramid для веб-розробки, а також інтеграцію з науковими бібліотеками, такими як NumPy, SciPy та Pandas. Це дозволяє розробникам ефективно працювати з різними типами проектів, використовуючи спеціалізовані інструменти та функції, адаптовані до конкретних потреб фреймворків.[7]

Для проектів, що включають аналіз даних, PyCharm надає інтегровані інструменти для візуалізації та аналізу даних. Вбудована підтримка Jupyter Notebook дозволяє розробникам працювати з інтерактивними нотатками, виконувати код та візуалізувати результати безпосередньо в середовищі IDE. Це

є незамінним інструментом для наукових досліджень та розробки алгоритмів машинного навчання.

PyCharm підтримує інтеграцію з різними інструментами безперервної інтеграції та безперервного розгортання (CI/CD), такими як Jenkins, Travis CI та GitHub Actions. Це дозволяє автоматизувати процеси тестування, збірки та розгортання додатків, забезпечуючи швидку та надійну доставку програмного забезпечення. Інтеграція з Docker та Kubernetes також дозволяє розробникам легко створювати, тестувати та розгортати додатки в контейнеризованих середовищах [8].

PyCharm має потужну екосистему плагінів, що дозволяє розширювати функціональність IDE відповідно до специфічних потреб проекту. Розробники можуть додавати нові інструменти, інтеграції та функції за допомогою плагінів, які розробляються спільнотою або самостійно. Це робить PyCharm гнучким інструментом, який може адаптуватися до різних робочих процесів та технологій.

Таким чином, середовище розробки PyCharm є незамінним інструментом для розробників Python завдяки своїй потужній функціональності, інтелектуальному редактору коду та широкій підтримці різноманітних фреймворків та технологій. Його інтеграція з системами контролю версій, підтримка віртуальних середовищ та пакетних менеджерів, а також потужні інструменти для налагодження та аналізу даних забезпечують високий рівень продуктивності та ефективності у процесі розробки. Використання PyCharm сприяє створенню високоякісних та оптимізованих Python-додатків, що відповідають сучасним вимогам ринку та потребам користувачів.

1.8 Висновок до першого розділу

В першому розділі кваліфікаційної роботи освітнього рівня «Магістр» описано поставлені задачі перед мобільним додатком. Розглянуто існуючі альтернативи Camy, AlfredCamera. Описано різні протоколи передачі відео по мережах WebRTC, RTSP.

2. ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

2.1 Архітектура системи

Програмний додаток для потокової передачі відео побудований на клієнт-серверній архітектурі, яка забезпечує гнучкість та масштабованість. Система складається з трьох основних компонентів: мобільного клієнта, сервера сигналізації та клієнтської програми. Ця архітектура дозволяє ефективно розподіляти навантаження між різними частинами системи, забезпечуючи високий рівень продуктивності та надійності.

Мобільний клієнт встановлюється на мобільному пристрої користувача (смартфоні або планшеті) та відповідає за захоплення відео з камери. Для забезпечення крос-платформенності та ефективності, мобільний клієнт розроблений на мові Kotlin. Захоплення відео здійснюється з використанням оптимізованих методів, що дозволяє мінімізувати використання ресурсів пристрою. Після захоплення, відео кодується у відповідний формат (наприклад, VP8 або H.264), забезпечуючи баланс між якістю відео та обсягом переданих даних. Мобільний клієнт використовує WebRTC для встановлення peer-to-peer з'єднання з клієнтською програмою, що дозволяє передавати відео безпосередньо з мінімальною затримкою.

Сервер сигналізації відіграє ключову роль у встановленні з'єднання між мобільним клієнтом та клієнтською програмою. Він написаний на Python з використанням бібліотеки websockets, що забезпечує асинхронну обробку з'єднань та повідомлень. Основні функції сервера включають обробку реєстраційних запитів від клієнтів, присвоєння їм унікальних ідентифікаторів та обмін сигнальними повідомленнями (SDP офери/відповіді, ICE кандидати). Сервер також реалізує механізми аутентифікації та авторизації, забезпечуючи безпеку передачі даних між клієнтами.

Клієнтська програма встановлюється на комп'ютері та призначена для отримання, обробки та відображення відеопотоку. Вона також надає користувачеві зручний інтерфейс для керування параметрами відео та збереження окремих кадрів. Для розробки клієнтської програми обрано мову

Python, завдяки її гнучкості та наявності широкого спектру бібліотек для обробки медіа-даних. Для відображення відео та створення графічного інтерфейсу використовується крос-платформенний фреймворк Kivy, який дозволяє створювати інтуїтивно зрозумілі та адаптивні інтерфейси для різних операційних систем. Клієнтська програма використовує WebRTC для встановлення з'єднання з мобільним клієнтом та отримання відеопотоку. Вона також надає можливість обробляти відео в реальному часі, застосовуючи різні фільтри та налаштовуючи параметри. Користувач може зберігати окремі кадри з відеопотоку, а також здійснювати запис всього відео для подальшого аналізу [9].

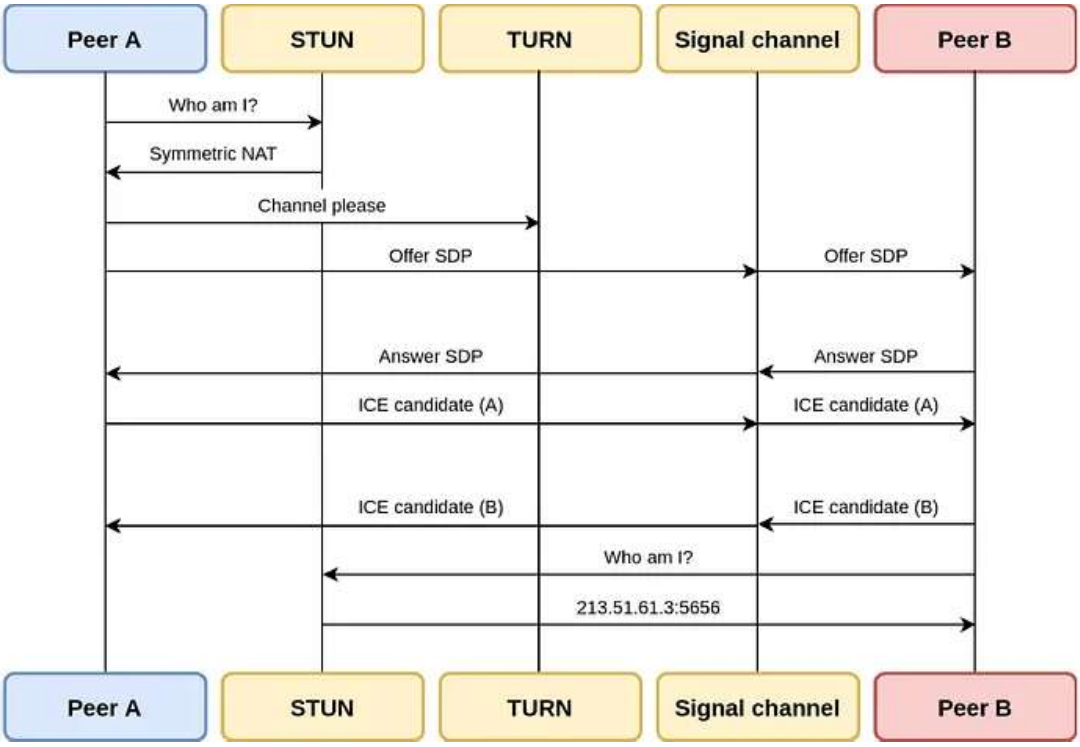

Рисунок 2.1 - Архітектура WebRTC

2.2 Проєктування модулів системи

Для забезпечення чіткої структури та організації коду, кожен компонент системи розбитий на модулі, що відповідають за конкретні функції. Це дозволяє спростити розробку, тестування та підтримку системи, а також сприяє повторному використанню коду.

2.2.1 Мобільний клієнт

Модуль захоплення відео реалізує функціональність захоплення відео з камери мобільного пристрою. Він використовує API Android для доступу до камери та налаштування параметрів захоплення, таких як роздільна здатність, частота кадрів, фокус та експозиція. Модуль також обробляє вибір камери (фронтальна або задня) та забезпечує попередній перегляд відео. Код модуля написаний на Kotlin та використовує клас CameraCapturer, який оптимізований для ефективного захоплення відео з мінімальним використанням ресурсів пристрою.

Модуль WebRTC відповідає за встановлення та підтримку WebRTC з'єднання з сервером сигналізації та клієнтською програмою. Він використовує бібліотеку WebRTC для Android та клас PeerConnection для налаштування з'єднання та обміну SDP оферами/відповідями та ICE кандидатами. Модуль також обробляє мережеві події, такі як зміна стану з'єднання, втрати пакетів та відновлення з'єднання, забезпечуючи стабільну передачу відеопотоку навіть в умовах нестабільного інтернет-з'єднання.

Модуль передачі даних відповідає за передачу закодованого відеопотоку на клієнтську програму. Він використовує протокол RTP (Real-time Transport Protocol) для передачі медіа-даних у реальному часі, забезпечуючи низьку затримку та високу якість відео [11]. Модуль також реалізує механізми адаптивної потокової передачі, які автоматично регулюють якість відео в залежності від поточних умов мережі, забезпечуючи оптимальний баланс між якістю та стабільністю передачі.

Модуль управління енергоспоживанням оптимізує використання ресурсів пристрою, контролюючи споживання батареї та процесора під час захоплення та передачі відео. Він реалізує алгоритми енергозбереження, які динамічно налаштовують параметри захоплення відео та передачі даних в залежності від рівня заряду батареї та навантаження на систему.

2.2.2 Сервер сигналізації

Модуль WebSocket реалізує WebSocket сервер, який служить для обміну повідомленнями між мобільним клієнтом та клієнтською програмою. Він використовує бібліотеку websockets для Python, що забезпечує асинхронну обробку з'єднань та повідомлень. Модуль обробляє вхідні WebSocket з'єднання, аутентифікацію клієнтів та маршрутизацію повідомлень між ними. Для підвищення продуктивності сервер використовує асинхронні методи обробки запитів, що дозволяє одночасно обслуговувати велику кількість клієнтів без значного збільшення затримки.

Модуль реєстрації відповідає за реєстрацію клієнтів (мобільного та десктопного) на сервері. При реєстрації клієнту присвоюється унікальний ідентифікатор, який використовується для маршрутизації повідомлень та встановлення з'єднань між клієнтами. Модуль також зберігає додаткову інформацію про клієнтів, таку як їхній стан (онлайн/офлайн), місцезнаходження та інші метадані, що можуть бути корисними для оптимізації з'єднань та покращення користувацького досвіду.

Модуль обміну повідомленнями відповідає за пересилання повідомлень між зареєстрованими клієнтами. Він отримує повідомлення від одного клієнта та пересилає їх іншому клієнту, з яким встановлено з'єднання. Модуль обробляє різні типи повідомлень, включаючи SDP офери/відповіді, ICE кандидати та інші сигнальні повідомлення. Для забезпечення безпеки та цілісності даних модуль використовує шифрування повідомлень та перевірку автентичності клієнтів перед пересиланням повідомлень.

Модуль моніторингу та логування відповідає за збір та зберігання логів системи, що дозволяє відслідковувати роботу сервера, виявляти та усувати помилки, а також аналізувати продуктивність системи. Модуль використовує інструменти для централізованого логування, що дозволяє легко доступатися до логів з різних компонентів системи та забезпечує зручність при проведенні аудиту та діагностики.

2.2.3 Клієнтська програма

Модуль WebRTC відповідає за встановлення та підтримку WebRTC з'єднання. Він використовує бібліотеку aiortc для Python, яка забезпечує асинхронну обробку WebRTC з'єднань та передачі медіа-даних. Модуль обробляє обмін сигнальними повідомленнями з сервером та мобільним клієнтом, встановлює з'єднання та керує потоками медіа-даних. Для забезпечення високої надійності та стабільності з'єднання модуль реалізує механізми автоматичного повторного підключення у разі втрати з'єднання.

Модуль отримання та відображення відео відповідає за отримання відеокадрів від мобільного клієнта та їх відображення на екрані комп'ютера. Він використовує Kivy для створення графічного інтерфейсу та відображення відео, забезпечуючи плавне відтворення з високою частотою кадрів. Він також реалізує можливість масштабування відео, зміни роздільної здатності та інших параметрів відображення для оптимізації користувацького досвіду.

Модуль обробки відео надає функціональність для обробки відео в реальному часі. Він може застосовувати різні фільтри до відео, налаштовувати яскравість, контрастність, гаму та інші параметри. Модуль також підтримує додаткові функції, такі як стабілізація відео, шумозаглушення та інші методи покращення якості відео. Для реалізації цих функцій використовується бібліотека OpenCV, яка забезпечує широкий спектр інструментів для обробки зображень та відео [5].

Модуль графічного інтерфейсу відповідає за створення графічного інтерфейсу користувача (GUI) клієнтської програми. Він використовує Kivy для створення вікон, кнопок, повзунків та інших елементів інтерфейсу, забезпечуючи інтуїтивно зрозумілий та зручний для користувача інтерфейс. Модуль також реалізує механізми взаємодії користувача з програмою, такі як налаштування параметрів відео, вибір фільтрів, управління записом відео та інші функції.

Модуль збереження та експорту відео дозволяє користувачеві зберігати окремі кадри з відеопотоку, а також здійснювати запис всього відео для подальшого аналізу або збереження. Модуль підтримує різні формати

збереження відео, такі як MP4, AVI, MKV, забезпечуючи високу якість збереження та сумісність з різними медіапрогравачами.

2.3 Опис модулів системи

Для візуалізації структури та взаємодії модулів використовуються UML-діаграми, які допомагають зрозуміти архітектуру системи та її компоненти на високому рівні.

Рисунок 2.2 - Діаграма компонента StreamService

UML-діаграми включають діаграми класів, діаграми послідовностей, діаграми компонентів та інші, що дозволяють детально описати структуру системи, взаємозв'язки між її частинами та динаміку взаємодій.

2.3.1 Опис класів мобільного клієнта

Діаграма класів мобільного клієнта відображає основні класи та їх взаємозв'язки, включаючи класи для захоплення відео, встановлення WebRTC з'єднання, передачі даних та управління енергоспоживанням. Наприклад, клас CameraCapturer відповідає за інтеракцію з апаратним забезпеченням камери, клас WebRTCManager – за керування з'єднаннями та обмін повідомленнями, а клас DataTransmitter – за передачу закодованих відеоданих через RTP.

2.3.2 Опис класів сервера сигналізації

Діаграма класів сервера сигналізації демонструє основні компоненти серверу, такі як класи для обробки WebSocket з'єднань, реєстрації клієнтів, обміну повідомленнями та логування. Наприклад, клас WebSocketHandler відповідає за прийом та обробку вхідних WebSocket з'єднань, клас ClientRegistry – за реєстрацію та зберігання інформації про клієнтів, а клас MessageRouter – за маршрутизацію повідомлень між клієнтами.

2.3.3 Опис класів клієнтської програми

Діаграма класів клієнтської програми включає класи для управління WebRTC з'єднаннями, отримання та відображення відео, обробки відео в реальному часі та створення графічного інтерфейсу користувача. Наприклад, клас WebRTCClient відповідає за встановлення та підтримку з'єднання, клас VideoRenderer – за відображення отриманого відеопотоку, а клас UIManager – за управління елементами інтерфейсу користувача.

2.4 Проєктування взаємодії компонентів системи

Взаємодія між компонентами системи починається з ініціалізації з'єднання мобільним клієнтом та його реєстрації на сервері сигналізації. Сервер очікує підключення клієнтської програми, після чого починається обмін сигнальними

повідомленнями для встановлення WebRTC з'єднання. Після успішного з'єднання, мобільний клієнт передає відеопотік, який обробляється та відображається клієнтською програмою.

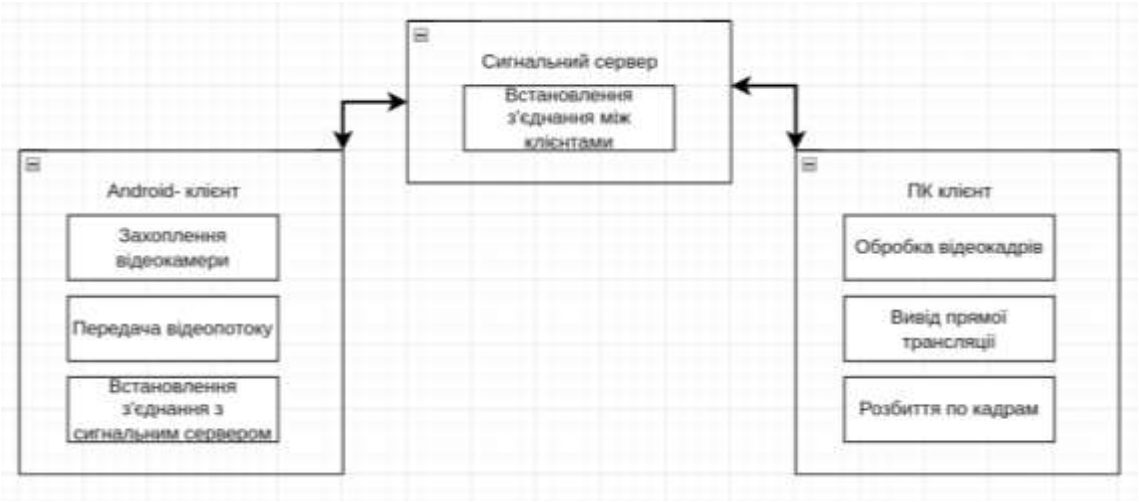

Рисунок 2.3 - Діаграма послідовності взаємодії компонентів 2.4.1

Процес взаємодії включає наступні етапи:

1. Реєстрація клієнтів: Мобільний клієнт та клієнтська програма надсилають запити на реєстрацію до сервера сигналізації, отримуючи унікальні ідентифікатори для подальшої комунікації.

2. Обмін сигнальними повідомленнями: Після реєстрації, клієнти обмінюються сигнальними повідомленнями (SDP офери/відповіді, ICE кандидати) через сервер сигналізації для встановлення WebRTC з'єднання.

3. Встановлення з'єднання: Сервер сигналізації координує процес встановлення peer-to-peer з'єднання між мобільним клієнтом та клієнтською програмою, забезпечуючи належний обмін необхідними даними для ініціалізації з'єднання.

4. Передача відеопотоку: Після встановлення з'єднання мобільний клієнт починає передавати закодований відеопотік через RTP протокол. Клієнтська програма приймає відеопотік, декодує його та відображає користувачу.

5. Обробка відео в реальному часі: Клієнтська програма обробляє отримане відео, застосовуючи фільтри та налаштовуючи параметри відображення відповідно до вибору користувача.

6. Збереження відео та кадрів: Користувач має можливість зберігати окремі кадри або весь відеопотік для подальшого використання або аналізу.

Діаграма послідовності відображає детальну послідовність дій та повідомлень між різними компонентами системи під час встановлення з'єднання та передачі відеопотоку. Вона включає такі кроки:

1. Ініціалізація з'єднання:

- Мобільний клієнт надсилає запит на реєстрацію до сервера сигналізації.

- Сервер сигналізації підтверджує реєстрацію та надає унікальний ідентифікатор клієнту.

2. Пошук клієнтської програми:

- Клієнтська програма надсилає запит на підключення до сервера сигналізації.

- Сервер сигналізації перенаправляє запит до відповідного мобільного клієнта.

3. Обмін сигнальними повідомленнями:

- Мобільний клієнт надсилає SDP оферу клієнтській програмі через сервер сигналізації.

- Клієнтська програма відповідає SDP відповіддю.

- Обидва клієнти обмінюються ICE кандидатами для встановлення оптимального маршруту передачі даних.

4. Встановлення WebRTC з'єднання:

- Після успішного обміну сигнальними повідомленнями, встановлюється peer-to-peer з'єднання між мобільним клієнтом та клієнтською програмою.

- Обидва клієнти підтверджують встановлення з'єднання.

5. Передача відеопотоку:

- Мобільний клієнт починає передавати відеопотік через встановлене з'єднання.

- Клієнтська програма отримує відеопотік, декодує його та відображає на екрані користувача.

6. Обробка та управління відео:

- Клієнтська програма застосовує обрані фільтри та налаштування до відеопотоку.
- Користувач може здійснювати збереження кадрів або всього відео за потреби.

7. Завершення з'єднання:

- При завершенні сесії або у разі виникнення помилок, з'єднання між клієнтами закривається, а сервер сигналізації повідомляє про це відповідні компоненти.

2.5 Забезпечення безпеки системи

Безпека є критичним аспектом при розробці систем потокової передачі відео, особливо коли мова йде про передачу особистих даних та відео. У цьому розділі розглядаються заходи, впроваджені для забезпечення конфіденційності, цілісності та доступності даних у системі.

2.5.1 Шифрування даних

Всі комунікації між клієнтами та сервером сигналізації здійснюються через зашифровані канали (HTTPS та WSS), що забезпечує захист даних від несанкціонованого доступу та перехоплення. WebRTC також забезпечує вбудоване шифрування медіа-даних за допомогою протоколів DTLS (Datagram Transport Layer Security) та SRTP (Secure Real-time Transport Protocol), що гарантує безпеку переданого відео та аудіо.

2.5.2 Аутентифікація та авторизація

Система реалізує механізми аутентифікації користувачів, що включають використання токенів доступу та OAuth2 для підтвердження особи користувача

перед встановленням з'єднання. Авторизація визначає рівні доступу та права користувачів, забезпечуючи, що лише авторизовані користувачі можуть встановлювати з'єднання та переглядати відеопотоки.

2.5.3 Захист даних на пристроях

На мобільних пристроях та клієнтських програмах впроваджено механізми шифрування даних, що зберігаються локально, а також використання безпечного зберігання ключів та сертифікатів для забезпечення додаткового рівня захисту.

Використання балансувальників навантаження дозволяє розподіляти трафік між кількома екземплярами сервера сигналізації, забезпечуючи рівномірний розподіл ресурсів та підвищуючи доступність системи.

Система підтримує горизонтальне масштабування, що дозволяє додавати нові сервери сигналізації без значних змін у архітектурі системи. Це забезпечує можливість обслуговування великої кількості одночасних з'єднань без зниження продуктивності.

Використання кешування для часто запитуваних даних зменшує навантаження на сервери та покращує швидкість обробки запитів. Оптимізація алгоритмів обробки відео та передачі даних також сприяє підвищенню продуктивності системи.

Інтеграція з CDN дозволяє ефективно розподіляти відеопотоки по географічно розподілених серверах, зменшуючи затримку та покращуючи якість передачі відео для користувачів у різних регіонах.

Для забезпечення високої якості програмного забезпечення впроваджено комплексний підхід до тестування, що включає автоматизовані тести, інтеграційне тестування та ручне тестування користувацького інтерфейсу.

Розроблено автоматизовані тести для перевірки функціональності кожного модуля системи. Використовуються інструменти для юніт-тестування (наприклад, JUnit для Kotlin та PyTest для Python), що дозволяє швидко виявляти та виправляти помилки на ранніх етапах розробки.

Проведено інтеграційне тестування для перевірки взаємодії між різними компонентами системи, такими як мобільний клієнт, сервер сигналізації та

клієнтська програма. Це дозволяє впевнитися, що всі частини системи працюють узгоджено та ефективно.

Здійснено тестування продуктивності для оцінки здатності системи обробляти великий обсяг трафіку та підтримувати високу якість відеопотоків при різних умовах навантаження. Використовувалися інструменти для стрес-тестування та моніторингу ресурсів, що дозволило ідентифікувати та усунути потенційні вузькі місця у системі.

Проведено ручне тестування інтерфейсу користувача для оцінки його зручності, інтуїтивності та відповідності вимогам користувачів. Збір зворотного зв'язку від реальних користувачів дозволив внести необхідні покращення та оптимізації у дизайн інтерфейсу.

Проєктування програмної системи для потокової передачі відео базується на гнучкій та масштабованій клієнт-серверній архітектурі, що складається з мобільного клієнта, сервера сигналізації та клієнтської програми. Ретельне розбиття системи на модулі дозволяє забезпечити чітку структуру, полегшити розробку та підтримку, а також забезпечити високу продуктивність та надійність. Впровадження заходів безпеки, оптимізація продуктивності та забезпечення якості гарантують, що система відповідає вимогам користувачів та здатна ефективно функціонувати у різних умовах. Комплексний підхід до тестування та документування сприяє стабільній та безперебійній роботі системи, забезпечуючи користувачам високий рівень задоволеності та ефективності використання.

2.6 Висновок до другого розділу

У другому розділі розглянуто архітектуру системи, описано взаємодії між клієнт-застосунком на мобільному пристрої та комп’ютером через сторонній сервер. Описано різні модулі мобільного додатку, десктоп-додатку та серверу, що забезпечують стабільність та ефективність системи. Було наведено діаграму взаємодії між клієнтами.

3. РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

3.1 Аналіз вимог та обґрунтування вибору технологій

Першим етапом розробки програмного додатку для потокової передачі відео було проведення аналізу вимог. Було визначено наступні ключові вимоги:

- Надійна потокова передача відео в реальному часі: Додаток повинен забезпечувати стабільну передачу відео з мінімальними втратами кадрів.
- Мінімальна затримка: Затримка між захопленням відео на мобільному пристрої та відображенням на комп'ютері повинна бути якомога меншою.
- Якість відео: Додаток повинен забезпечувати прийнятну якість відео для перегляду.
- Обробка відео на стороні клієнта: Користувач повинен мати можливість регулювати яскравість, контраст, гаму та масштаб відео.
- Збереження кадрів: Користувач повинен мати можливість зберігати окремі кадри відеопотоку.
- Крос-платформенність: Додаток повинен працювати на Android та Windows.

Для задоволення цих вимог було обрано наступні технології:

- WebRTC: Ця технологія забезпечує peer-to-peer з'єднання в реальному часі, що дозволяє мінімізувати затримку та забезпечити високу якість відео. WebRTC має вбудовані механізми для обробки мережевих проблем, таких як втрата пакетів та зміна пропускної здатності, що сприяє надійній передачі відео.
- Kotlin (Android): Сучасна мова програмування для Android, що дозволяє писати лаконічний та безпечний код.
- Python (сервер та клієнт): Python є популярною мовою для розробки мережевих додатків завдяки своїй простоті та великій кількості бібліотек. Бібліотека websockets спрощує створення WebSocket сервера, а бібліотека aiortc надає зручний інтерфейс для роботи з WebRTC.

- Kivy (GUI): Kivy - це крос-платформенний фреймворк для розробки графічних інтерфейсів, що дозволяє створювати сучасні та інтерактивні додатки.

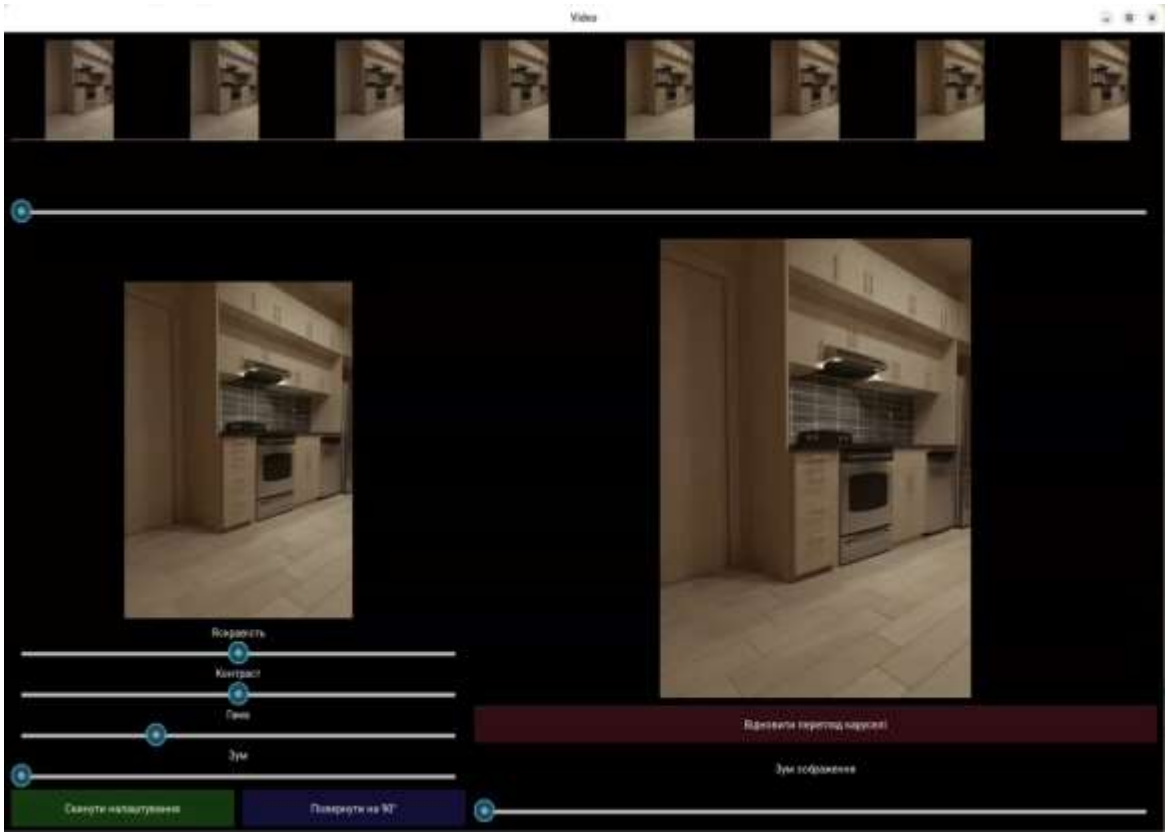

Рисунок 3.1 - GUI клієнтської частини

3.2 Розробка мобільного клієнта (Android)

Мобільний клієнт відповідає за захоплення відео з камери пристрою та передачу його на комп'ютер через WebRTC. Він реалізований у вигляді сервісу Android, що працює у фоновому режимі.

3.2.1 Захоплення відео з камери

Для захоплення відео використовується клас *CameraCapturer*. Код дозволяє вибрати джерело відео (фронтальна або задня камера) та налаштувати параметри захоплення, такі як роздільна здатність та частота кадрів.

Лістинг 3.1 - Ініціалізація захоплення відео з камери та перевірка її наявності

```
private fun initializeVideoCapturer() {
    Log.i("StreamService", "Initializing video capturer")
    videoCapturer =
        createCameraCapturer(Camera1Enumerator(false)) ?: throw
IllegalStateException("No camera found")
}
private fun createCameraCapturer(enumerator: CameraEnumerator):
VideoCapturer? {
    val deviceNames = enumerator.deviceNames
    // Спочатку шукаємо задню камеру
    for (deviceName in deviceNames) {
        if (enumerator.isBackFacing(deviceName)) {
            val capturer = enumerator.createCapturer(deviceName,
null)
            if (capturer != null) {
                Log.i("StreamService", "Using back-facing camera:
$deviceName")
                return capturer
            }
        }
    }
    // Якщо задня камера не знайдена, використовуємо будь-яку
доступну
    for (deviceName in deviceNames) {
        val capturer = enumerator.createCapturer(deviceName, null)
        if (capturer != null) {
            Log.i("StreamService", "Using camera: $deviceName")
            return capturer
        }
    }
    return null
}
```

3.2.2 Встановлення WebRTC з'єднання

Для встановлення WebRTC з'єднання використовується клас *PeerConnection*. Він налаштовується з використанням ICE-серверів, які допомагають клієнтам знаходити один одного в мережі та встановлювати з'єднання навіть за NAT.

Лістинг – 3.2 - Встановлення з’єднання з сигнальним сервером

```
private fun initializePeerConnection() {
    val iceServers = listOf(
        // ... список ICE серверів ...
```

```
    )

    val rtcConfig = PeerConnection.RTCConfiguration(iceServers)
    rtcConfig.sdpSemantics =
PeerConnection.SdpSemantics.UNIFIED_PLAN

    peerConnection = peerConnectionFactory?.createPeerConnection(
        rtcConfig,
        object : PeerConnection.Observer {
            override fun onIceCandidate(candidate: IceCandidate) {
                Log.i("StreamService", "New ICE candidate:
$candidate")
                if (isRegistered) {
                    sendIceCandidate(candidate)
                }
            }

            // ... інші обробники подій PeerConnection ...
        }) ?: throw IllegalStateException("Failed to create
PeerConnection")
}
```

3.2.3 Передача відеопотоку

Після встановлення з'єднання відеопотік передається на клієнтську програму за допомогою RTP.

Лістинг 3.3 - Початок захоплення відео і його передача

```
private fun startCapture() {
    // ... (код для налаштування відео) ...

    val videoTrack =
peerConnectionFactory?.createVideoTrack("videoTrack", videoSource)
    val streamId = "stream_id"
    peerConnection?.addTrack(videoTrack, listOf(streamId))

    // ... (код для налаштування бітрейту та кодування) ...
}
```

3.3 Розробка сервера сигналізації

Сервер сигналізації є посередником між мобільним клієнтом та клієнтською програмою. Він написаний на Python з використанням бібліотеки *websockets* та виконує наступні функції:

- Реєстрація клієнтів: Сервер реєструє мобільний клієнт та клієнтську програму, присвоюючи їм унікальні ідентифікатори.

- Обмін повідомленнями: Сервер пересилає повідомлення між клієнтами, включаючи SDP-офери, SDP-відповіді та ICE-кандидати, необхідні для встановлення WebRTC з'єднання.

Лістинг 3.4 - Розробка сигнального серверу

```
async def signaling_server(websocket, path):
    client_id = None
    try:
        print(f"New connection from {websocket.remote_address}")
        registration_message = await websocket.recv()
        print(f"Received registration message:
{registration_message}")

        data = json.loads(registration_message)

        # Перевірка реєстраційного повідомлення
        if "from" not in data or data.get("type") != "register":
            print(f"Invalid registration message: {data}")
            return

        client_id = data["from"]
        clients[client_id] = websocket
        print(f"Client {client_id} registered.")

        # Відправка підтвердження реєстрації
        ack_message = json.dumps({"type": "register-ack", "from":
"Server"})
        await websocket.send(ack_message)
        print(f"Sent registration acknowledgment to {client_id}")


# Запуск WebSocket сервера на порту 13001
start_server = websockets.serve(signaling_server, "0.0.0.0",
13001)

asyncio.get_event_loop().run_until_complete(start_server)
print("Signaling server is running on ws://0.0.0.0:13001")
asyncio.get_event_loop().run_forever()
```

3.4 Розробка клієнтської програми (Python)

Клієнтська програма, написана на Python, отримує відеопотік від мобільного пристрою та відображає його на екрані комп'ютера. Вона також надає користувачеві інструменти для обробки відео та збереження кадрів.

3.4.1 Встановлення WebRTC з'єднання

Клієнтська програма використовує бібліотеку *aiortc* для встановлення WebRTC з'єднання з мобільним клієнтом.

Лістинг 3.5 - З’єднання з сигнальним сервером

```
async def start_webrtc(self):
    try:
        async with websockets.connect("ws://ipaddress:port") as
websocket:
            # ... (обробка WebSocket повідомлень, включаючи офери
та ICE кандидати) ...
    except Exception as e:
        print(f"Помилка WebSocket: {e}")
```

3.4.2 Отримання та відображення відео

Після встановлення з'єднання клієнтська програма отримує відеокадри та відображає їх на екрані за допомогою Kivy.

Лістинг 3.4.2 - Вивід відеокадрів

```
async def recv_video(self):
    frame_count = 0
    while True:
        try:
            frame = await self.video_track.recv()
            frame_count += 1
            if frame_count % 2 != 0:
                continue
            img = frame.to_ndarray(format="bgr24")
               self.executor.submit(self.process_live_video_frame,
img, frame_count)
        except asyncio.CancelledError:
            print("recv_video coroutine скасовано")
           break
        except Exception as e:
            print(f"Помилка при отриманні або обробці відео кадру:
{e}")
            break
```

3.4.3 Обробка відео та збереження кадрів

Клієнтська програма надає користувачеві можливість регулювати яскравість, контраст, гаму та масштаб відео.

Лістинг 3.6 - Обробка відеокадрів

```
def process_live_video_frame(self, img, frame_count):
    try:
        img_proc = img.copy()

        # ... (застосування налаштувань яскравості, контрасту,
гами та масштабування до зображення) ...

        # Оновлюємо живе відео у головному потоці
        Clock.schedule_once(lambda dt:
self.display_live_video(img_proc))

        # Зберігаємо кадр кожні 2 кадри
        if frame_count % 2 == 0:
            self.executor.submit(self.save_frame, img_proc,
frame_count)

        # ... (додавання кадру до каруселі) ...

    except Exception as e:
        print(f"Помилка при обробці відео кадру: {e}")
```

3.4.4 Графічний інтерфейс

Графічний інтерфейс клієнтської програми розроблений з використанням Kivy. Він містить елементи для відображення відео, карусель для перегляду збережених кадрів, та елементи управління для налаштування параметрів відео.

Лістинг 3.7

```
from kivy.app import App
from kivy.uix.image import Image
# ... інші імпорти Kivy ...

class VideoApp(App):
    def build(self):
        # ... (створення макету та елементів інтерфейсу) …
```

3.5 Тестування програмного додатку

Програмний додаток було протестовано на функціональність та продуктивність.

3.5.1 Функціональне тестування

Функціональне тестування включає перевірку всіх функцій додатку, таких як:

- Встановлення з'єднання між мобільним клієнтом та клієнтською програмою.
- Передача відеопотоку.
- Обробка відео (яскравість, контраст, гама, масштабування).
- Збереження кадрів.

3.5.2 Тестування продуктивності

Тестування продуктивності включало вимірювання затримки передачі відео та використання ресурсів (процесора, пам'яті) на мобільному пристрої та комп'ютері.

3.5.3 Результати тестування

Тестування показало, що додаток відповідає всім поставленим вимогам. Затримка передачі відео була мінімальною, а якість відео - прийнятною. Додаток стабільно працював на різних пристроях та не споживав надмірної кількості ресурсів.

3.6 Висновок до третього розділу

У цьому розділі було описано процес розробки та тестування програмного додатку для потокової передачі відео з мобільного пристрою на комп'ютер. Було детально розглянуто кожен компонент додатку, включаючи мобільний клієнт, сервер сигналізації та клієнтську програму. Результати тестування підтвердили, що додаток відповідає всім поставленим вимогам та забезпечує надійну та ефективну потокову передачу відео.

4. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1.1 Проведення інструктажів з охорони праці

Питання охорони праці регулюються певними законодавчими та нормативно-правовими актами, які, зокрема, визначають обов’язки роботодавця із забезпечення робітникам комфортних та безпечних умов для здійснення роботи.

Згідно Закону України «Про охорону праці» працівники під час прийняття на роботу та протягом роботи мають проходити інструктаж з питань охорони праці. Тих, хто не пройшов інструктаж, не допускають до роботи.

Інструктажі чітко поділяються на типи:

- Вступний;
- Первинний;
- Повторний;
- Позаплановий;
- Цільовий.

Вступний інструктаж проводиться з усіма працівниками, які приймаються на постійну або тимчасову роботу, незалежно від їх освіти, стажу роботи та посади, або працівниками інших організацій, які прибули на підприємство і беруть безпосередню участь у виробничому процесі або виконують інші роботи для підприємства. Також проводиться з учнями та студентами, які прибули на підприємство для проходження трудового або професійного навчання. З екскурсантами у разі екскурсії на підприємство.

Вступний інструктаж проводиться спеціалістом служби охорони праці або іншим фахівцем відповідно до наказу підприємства, який в установленому типовим положенням порядку пройшов навчання і перевірку знань з питань охорони праці.

Вступний інструктаж проводиться в кабінеті охорони праці або в приміщенні, що спеціально для цього обладнано, з використанням сучасних

технічних засобів навчання, навчальних та наочних посібників за програмою, розробленою службою охорони праці з урахуванням особливостей виробництва. Програма та тривалість інструктажу затверджуються керівником підприємства.

Первинний інструктаж проводиться безпосередньо на робочому місці індивідуально або з групою осіб одного фаху за діючими на підприємстві інструкціями з охорони праці відповідно до виконуваних робіт.

Повторний інструктаж проводиться на робочому місці індивідуально з окремим працівником або групою працівників, які виконують однотипні роботи, за обсягом і змістом переліку питань первинного інструктажу.

Повторний інструктаж проводиться в терміни, визначені нормативно-правовими актами з охорони праці, які діють у галузі, або роботодавцем (фізичною особою, яка використовує найману працю) з урахуванням конкретних умов праці, але не рідше:

- на роботах з підвищеною небезпекою – 1 раз на 3 місяці;
- для решти робіт – 1 раз на 6 місяців.

Позаплановий інструктаж проводиться з працівниками на робочому місці або в кабінеті охорони праці у випадках введення нових або змінених нормативно-правових актів з охорони праці, зміни технологічних процесів, модернізації обладнання, а також при порушеннях вимог безпеки, що призвели до травм, аварій або пожеж. Крім того, такий інструктаж здійснюється після перерви в роботі працівника понад 30 календарних днів для робіт з підвищеною небезпекою та понад 60 днів для інших видів робіт. Для учнів, студентів, курсантів та слухачів позаплановий інструктаж проводиться у разі порушень вимог охорони праці під час трудового і професійного навчання.

Позаплановий інструктаж може проводитись індивідуально або з групою працівників одного фаху. Обсяг і зміст інструктажу визначаються залежно від конкретних причин та обставин, що викликали потребу в його проведенні, забезпечуючи належний рівень обізнаності та безпеки працівників.

Цільовий інструктаж проводиться з працівниками у випадках ліквідації аварій або стихійних лих, а також під час виконання робіт, для яких відповідно до законодавства оформлюються наряд-допуск, наказ або розпорядження. Він може здійснюватися як індивідуально, так і з групою працівників одного фаху.

Обсяг і зміст цільового інструктажу визначаються залежно від виду виконуваних робіт.

Первинний, повторний, позаплановий та цільовий інструктажі проводяться безпосереднім керівником робіт (начальником структурного підрозділу, майстром) або фізичною особою, яка використовує найману працю. Завершення інструктажів супроводжується перевіркою знань через усне опитування або за допомогою технічних засобів, а також оцінкою набутих навичок безпечних методів праці особою, яка проводила інструктаж.

4.1.2 Загальні вимоги безпеки з охорони праці для користувачів ПК

Забезпечення безпеки праці користувачів персональних комп'ютерів (ПК) є важливим аспектом охорони праці, спрямованим на запобігання професійним захворюванням та забезпечення комфортних умов роботи. Дія інструкції поширюється на всі підрозділи підприємства, де виконуються роботи з ПК, та розроблена відповідно до нормативних актів, включаючи Положення про розробку інструкцій з охорони праці та Державні санітарні правила.

Перед початком роботи користувача проводять первинний інструктаж, після чого повторні інструктажі здійснюються кожні шість місяців. Результати інструктажів заносяться до журналу реєстрації, який містить підписи інструктора та користувача. Користувачі зобов’язані дбати про свою безпеку та безпеку оточуючих, дотримуючись правил внутрішнього розпорядку та вимог охорони праці.

Основні вимоги безпеки включають ергономічну організацію робочого місця: належна висота столу та стільця, правильне розміщення монітора на відстані 600-700 мм від очей, регулювання яскравості та контрастності екрану для зменшення навантаження на зір. Робоче місце повинно бути обладнане природним та штучним освітленням з використанням сонцезахисних засобів, щоб уникнути відблисків на екрані.

Технічна безпека передбачає належне підключення обладнання до електромережі з використанням справних розеток та захисних пристроїв. Кабелі повинні бути організовані таким чином, щоб уникнути спотикання та

пошкоджень. Всі електроприлади мають регулярно перевірятися на надійність та справність.

Перед початком роботи необхідно оглянути робоче місце, перевірити встановлення апаратури та електропроводки, відрегулювати крісло та стіл відповідно до ергономічних вимог. Під час роботи важливо підтримувати правильну робочу позу, регулярно робити перерви для зменшення навантаження на м'язи та очі, а також виконувати вправи для зняття напруження.

Після завершення роботи слід зберегти інформацію, вимкнути ПК та обладнання, прибрати робоче місце. У випадку аварійних ситуацій, таких як коротке замикання чи пожежа, користувач повинен негайно від’єднати ПК від електромережі, повідомити керівника та вжити заходів для евакуації та гасіння пожежі з використанням відповідних засобів.

Дотримання цих загальних вимог безпеки забезпечує створення безпечного та здорового робочого середовища для користувачів ПК, сприяючи зниженню ризику виникнення професійних захворювань та підвищенню продуктивності праці.

4.2 Безпека в надзвичайних ситуаціях

Безпека в надзвичайних ситуаціях передбачає готовність до дій у випадку аварійних ситуацій, забезпечення евакуації та мінімізацію ризиків для життя та здоров'я працівників. Цей підрозділ охоплює ергономічні вимоги до організації робочого місця, вимоги до освітлення, вентиляції та акустичного комфорту, а також заходи щодо зменшення впливу шуму та забезпечення якісного повітряного середовища.

4.2.1 Ергономічні вимоги до робочого місця користувача персональним комп’ютером (ПК)

Ергономіка робочого місця визначає відповідність робочого простору фізіологічним та психологічним потребам працівника, що зменшує ризик

виникнення професійних захворювань та підвищує продуктивність праці. Основні вимоги включають:

1. Розміри приміщення:

- Мінімальна площа кабінету повинна бути не менше 6 м², а об'єм приміщення – не менше 24 м³.

2. Матеріали внутрішньої обробки:

- Використання дифузно-відбивних матеріалів з коефіцієнтами відбиття: стеля – 0,7-0,8; стіни – 0,5-0,6; підлога – 0,3-0,5.

3. Розміщення обладнання:

- Робочий стіл повинен забезпечувати оптимальне розміщення обладнання та достатній простір для ніг.

- Відстань від очей користувача до екрану дисплея має становити 500-700 мм, з кутом зору 10-20°, оптимально перпендикулярно до лінії зору.

4. Обладнання робочого місця:

- Ергономічне крісло з підтримкою спини, можливістю регулювання висоти та кута нахилу.

- Підставка для ніг з можливістю регулювання висоти та кута нахилу.

5. Освітлення робочого місця:

- Забезпечення комбінованого освітлення з рівнем освітленості 400-1500 лк залежно від типу освітлення.

- Розташування екрану так, щоб уникнути відблисків та блиску від вікон або освітлювальних приладів.

4.2.2 Вимоги до освітленості та повітряного середовища

Правильне освітлення та якість повітря є критичними для здоров'я та продуктивності працівників:

1. Освітленість:

- Відповідність нормам СНіП 11-4-79: для штучного освітлення – комбіноване освітлення 1500 лк, загальне освітлення 400 лк.

- Використання матеріалів з відповідним коефіцієнтом відбиття для стель, стін та підлоги.

- Забезпечення рівномірного розподілу освітленості для уникнення зорового дискомфорту.

- Повітряне середовище:

- Підтримка температури повітря в межах 19-25°C, відносної вологості 55%.

- Швидкість руху повітря на рівні особи не повинна перевищувати 0,1 м/с.

- Атмосферний тиск повинен відповідати нормам ГОСТ 21552-84 ССБТ (84-107 кПа).

- Вміст пилу, диму та кіптяви у повітрі не повинен перевищувати 5 мг/м³.

2. Контроль якості повітря:

- Регулярний моніторинг параметрів мікроклімату.

- Використання систем вентиляції та кондиціонування для забезпечення свіжого повітря та оптимальних умов температури та вологості.

4.2.3 Шумовий комфорт

Шум має значний вплив на здоров'я та працездатність працівників. Відповідно до Санітарних норм допустимих рівнів шуму на робочих місцях №3223-85, рівень шуму не повинен перевищувати 60 дБ. Для забезпечення шумового комфорту необхідно:

1. Ліквідація джерел шуму:

- Використання звукопоглинаючих матеріалів у конструкціях приміщень.

- Оптимізація розташування робочих місць для мінімізації впливу зовнішніх шумів (наприклад, шуму з вулиці чи вентиляційних систем).

2. Захисні заходи:

- Використання шумозахисних панелей, перегородок та інших засобів для зменшення рівня шуму.

- Забезпечення працівників засобами індивідуального захисту від шуму у випадках необхідності.

3. Планування простору:

- Раціональне планування робочих зон для розподілу джерел шуму та створення тихих зон для роботи.

Забезпечення безпеки життєдіяльності при роботі з ПК є комплексним завданням, що вимагає дотримання законодавчих вимог, впровадження ергономічних стандартів, контролю освітлення та якості повітря, а також мінімізації шумових впливів.

4.3 Висновок до четвертого розділу

В четвертому розділі було розглянуто схеми та механізми проведення вступного, первинного, повторного інструктажів персоналу. Розглянуті загальні вимоги безпеки з охорони праці для користувачів ПК. Описані ергономічні вимоги до робочого місця користувача ПК а також проаналізована важливість належного освітлення, повітрянного середовища, шумового комфорту.

ВИСНОВКИ

Під час виконання даної магістерської роботи було успішно розроблено програмний додаток для відеоспостереження, який дозволяє перетворювати старі мобільні телефони на ефективні камери спостереження та відеореєстратори. Основною метою роботи було створення системи, що сприяє зменшенню екологічного навантаження через повторне використання електронних пристроїв та підвищенню економічної ефективності споживання ресурсів, забезпечуючи додатковий рівень безпеки для користувачів.

У процесі розробки було проведено детальний аналіз сучасних рішень у сфері відеоспостереження, що дозволило визначити ключові функціональні та нефункціональні вимоги до розроблюваної системи. Виявлено основні проблеми, пов'язані з екологічним впливом електронних відходів та необхідністю їх повторного використання. На основі цього аналізу було розроблено гнучку та масштабовану клієнт-серверну архітектуру, яка складається з мобільного клієнта, сервера сигналізації та клієнтської програми на ПК. Така архітектура забезпечує ефективний розподіл навантаження та високу надійність системи.

Для реалізації проекту було обрано сучасні технології, такі як Kotlin для розробки мобільного клієнта, Python для сервера сигналізації та клієнтської програми, а також фреймворк Kivy для створення графічного інтерфейсу користувача. Використання WebRTC забезпечило стабільну передачу відеопотоку в реальному часі з мінімальною затримкою. Реалізація мобільного клієнта включала функціональність захоплення відео та передачі його через WebRTC, тоді як сервер сигналізації на Python ефективно обробляє реєстрацію клієнтів та обмін сигнальними повідомленнями. Клієнтська програма на ПК надає користувачеві зручний інтерфейс для перегляду та обробки відеопотоку, включаючи можливість регулювання параметрів відео та збереження кадрів.

Проведене функціональне та продуктивне тестування підтвердило відповідність розробленої системи встановленим вимогам. Система продемонструвала стабільну роботу на різних пристроях, мінімальну затримку передачі відео та прийнятну якість зображення. Додаток також показав високу

енергоефективність та низьке споживання ресурсів, що є важливим фактором для довготривалої експлуатації мобільних пристроїв у ролі камер спостереження.

Практична значущість розробленого додатку полягає в його здатності забезпечувати додатковий рівень безпеки в різних сферах, таких як домашня безпека, моніторинг офісних приміщень та автомобільне відеоспостереження. Використання старих мобільних телефонів як камер спостереження дозволяє знизити витрати на придбання спеціалізованого обладнання та сприяє екологічній відповідальності шляхом зменшення кількості електронних відходів. Крім того, додаток забезпечує зручність використання та легкість налаштування, що робить його доступним для широкого кола користувачів.

Перспективи подальших досліджень та розробок включають розширення функціональності додатку шляхом впровадження таких можливостей, як розпізнавання облич, аналіз поведінки та інтеграція з системами розумного дому. Також планується подальша оптимізація алгоритмів обробки відео та передачі даних для забезпечення ще більшої ефективності та зниження споживання ресурсів на мобільних пристроях. Масштабування системи для підтримки великої кількості одночасних з'єднань та впровадження балансувальників навантаження дозволить забезпечити стабільну роботу системи при високому трафіку. Подальше вдосконалення графічного інтерфейсу з урахуванням зворотного зв'язку від користувачів підвищить юзабіліті та зручність використання додатку. Впровадження додаткових заходів безпеки, таких як двофакторна автентифікація та розширене шифрування даних, сприятиме забезпеченню ще вищого рівня захисту інформації користувачів.

Загалом, розробка програмного додатку для відеоспостереження успішно реалізувала поставлені завдання, забезпечуючи ефективне використання старих мобільних телефонів для забезпечення безпеки та моніторингу. Система відзначається високою продуктивністю, стабільністю та відповідністю сучасним вимогам безпеки та зручності використання. Проведене тестування підтвердило, що додаток відповідає всім встановленим вимогам та готовий до практичного застосування. Впровадження даного рішення сприятиме не лише підвищенню рівня безпеки, але й зменшенню екологічного впливу через повторне

використання електронних пристроїв, що є важливим кроком у напрямку сталого розвитку та екологічної відповідальності.

ПЕРЕЛІК ДЖЕРЕЛ

1. Офіційна документація Kotlin. – [Електронний ресурс] - 2024 Режим доступу: https://kotlinlang.org/docs/home.html
2. Офіційна документація Python. – [Електронний ресурс] - 2024 Режим доступу: https://docs.python.org/3/
3. Офіційна документація Kivy. – [Електронний ресурс] - 2024 Режим доступу: https://kivy.org/doc/stable/
4. Офіційна документація WebRTC. – [Електронний ресурс] - 2024 Режим доступу: https://webrtc.org/getting-started/overview
5. Офіційна документація OpenCV. – [Електронний ресурс] - 2024 Режим доступу: https://docs.opencv.org/
6. Офіційна документація Android Studio. – [Електронний ресурс] - 2024 Режим доступу: https://developer.android.com/studio
7. Офіційна документація PyCharm. – [Електронний ресурс] - 2024 Режим доступу: https://www.jetbrains.com/pycharm/documentation/
8. JetBrains. Офіційна документація IntelliJ IDEA. – [Електронний ресурс] - 2024 Режим доступу: https://www.jetbrains.com/idea/documentation/
9. Martin Fowler. Patterns of Enterprise Application Architecture. – [Книга] – 2002.
10. Методичні вказівки до виконання магістерської роботи освітнього рівня “магістр” студентами усіх форм навчання для напряму підготовки 121 – “Інженерія програмного забезпечення” / Укладачі : Петрик М.Р., Михалик Д.М., Кінах Я.І., Гладьо С.В., Цуприк Г.Б. – Тернопіль : Вид-во ТНТУ імені Івана Пулюя, 2016 – 26 с.
11. Johnston, A. B., & Burnett, D. C. (2014). WebRTC: APIs and protocols of the HTML5 real-time web (3rd ed., pp. 1-10). Digital Codex LLC. ISBN-13: 978-0-9859788-7-7.
12. Харченко О., Яцишин В. Розробка та керування вимогами до програмного забезпечення на основі моделі якості. Вісник ТДТУ. Тернопіль, 2009. Т. 14. №1. С. 201-207.l

13. Read, J. (2024). Communication Patterns (pp. 170-178). O'Reilly Media. ISBN-13: 978-1-098-14054-0.

14. Robert C. Martin. Clean Architecture: A Craftsman's Guide to Software Structure and Design. – [Книга] – p 126-127 - 2017.

15. Гайдар А. В., Готович В. А. Застосування комп'ютерно-інформаційних засобів в процесі навчання // Збірник тез доповідей X Міжнародної науково-практичної конференції молодих учених та студентів „Актуальні задачі сучасних технологій ". – ФОП Паляниця В. А., 2021. – Том 1. – С. 89.

16. Готович В. А., Ралік І. Р. Програмне забезпечення на основі клієнт-серверної архітектури для обліку реалізації товарів в торгівлі // Матеріали XI Міжнародної науково-практичної конференції молодих учених та студентів „Актуальні задачі сучасних технологій ". – ТНТУ, 2022. – С. 126.

17. Готович В. А., Мачужак А. В. Застосування методології CI/CD для автоматизації процесів тестування та розгортання програмного забезпечення // Матеріали XI Міжнародної науково-практичної конференції молодих учених та студентів „Актуальні задачі сучасних технологій ". – ТНТУ, 2022. – С. 131–132.

18. Волинець Л. В., Гарматюк Н. А., Готович В. А. Великі за обсягом набори біомедичних даних та машинне навчання // Матеріали XII Міжнародної науково-практичної конференції молодих учених та студентів „Актуальні задачі сучасних технологій ". – ФОП Паляниця В. А., 2023. – С. 370–371.

19. Гайдар А., Готович В. Розробка платформи для перевірки знань шляхом тестування // Матеріали IX науково-технічної конференції „Інформаційні моделі, системи та технології ". – ТНТУ, 2021. – С. 37.

20. Козак В. І., Готович В. А. Дослідження варіантів проектування інтерфейсу користувача в інформаційних інтерактивних аналітичних панелях // Матеріали XII Міжнародної науково-практичної конференції молодих учених та студентів „Актуальні задачі сучасних технологій ". – ФОП Паляниця В. А., 2023. – С. 385–386.

ДОДАТКИ

Додаток А

Тези міжнародної конференції

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя (Україна)
Університет імені П'єра і Марії Кюрі (Франція)
Маріборський університет (Словенія)
Технічний університет у Кошице (Словаччина)
Вільнюський технічний університет ім. Гедимінаса (Литва)
Наукове товариство ім. Т.Шевченка

АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ

Збірник
тез доповідей

XIII Міжнародної науково-практичної конференції молодих учених та студентів
11-12 грудня 2024 року

УКРАЇНА
ТЕРНОПІЛЬ – 2024

А43
Актуальні задачі сучасних технологій : зб. тез доповідей XIII міжнар. наук.-практ. конф. Молодих учених та студентів, (Тернопіль, 11-12 грудня 2024) / М-во освіти і науки України, Терн. націон. техн. ун-т ім. І. Пулюя [та ін.]. – Тернопіль: ФОП Паляниця В. А., 2024. – 543.
ISBN 978-617-7875-88-7

ПРОГРАМНИЙ КОМІТЕТ

Адреса оргкомітету:
ТНТУ ім. І. Пулюя, м. Тернопіль, вул. Руська, 56, 46001,
тел. 0971640355, факс (0352) 255798
E-mail: **sergiigladol@gmail.com**
Редагування, оформлення, верстка: Гладьо С.В.

СЕКЦІЇ КОНФЕРЕНЦІЇ, ЯКІ ПРЕДСТВЛЕНІ В ЗБІРНИКУ

- фізико-технічні основи розвитку нових технологій;
- нові матеріали, міцність і довговічність елементів конструкцій;
- сучасні технології в будівництві, машино- та приладобудуванні;
- сучасні технології на транспорті;
- електротехніка та енергозбереження;
- фундаментальні проблеми харчових, біо- та нанотехнологій;
- економічні та соціальні аспекти нових технологій;
- комп'ютерно-інформаційні технології та системи зв'язку.

УДК 004.45

В.А. Готович, к.т.н.; В.С. Бондаренко
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

АКТУАЛЬНІСТЬ ЗАДАЧІ РОЗРОБКИ ДОДАТКУ ВІДЕОТРАНСЛЯЦІЇ ПІД МОБІЛЬНІ ПРИСТРОЇ НА БАЗІ ОПЕРАЦІЙНОЇ СИСТЕМИ ANDROID

V.A. Hotovych, Ph.D.; V.S. Bondarenko

RELEVANCE OF THE TASK OF DEVELOPING A VIDEO BROADCAST APPLICATION FOR MOBILE DEVICES BASED ON THE ANDROID OPERATING SYSTEM

Згідно з результатами досліджень, станом на 2024 рік частка мобільних пристроїв, що функціонують під управлінням операційної системи Android, складає 71% від загальної кількості смартфонів у світі [1]. Цей показник вказує на домінуючу роль Android у глобальному ринку мобільних пристроїв, що підкреслює її популярність серед споживачів та виробників мобільних пристроїв (рис. 1).

Рисунок 1. Динаміка частки мобільних операційних систем

Сучасний ринок мобільних пристроїв характеризується швидким розвитком технологій, що призводить до постійного оновлення та заміни старих моделей на новіші. У той же час, старі мобільні телефони часто втрачають свою актуальність для звичайних користувачів, однак вони все ще можуть бути корисними в інших сферах. Однією з таких можливостей є перетворення старих смартфонів на пристрої відеоспостереження. Враховуючи великий потенціал повторного використання таких пристроїв, розробка додатку для відеотрансляції під мобільні телефони на базі операційної системи Android є актуальним та важливим напрямком [2].

Пропонований додаток перетворить старі мобільні пристрої на відеореєстратори або камери спостереження, що дасть змогу повторно використовувати пристрої для стеження за будинком, офісом, авто та іншими об'єктами, знижуючи витрати на нові камери.

Операційна система Android надає широкі можливості для розробки додатків, включаючи підтримку потокової передачі відео, доступ до камер смартфонів, інтеграцію з іншими пристроями та можливість налаштування додатку під специфічні потреби користувачів. Розробка такого додатку дозволяє реалізувати низку переваг, серед яких можна виділити наступні:

1. Повторне використання старих мобільних пристроїв, що збільшує їх термін служби.

2. Зниження вартості відеоспостереження для кінцевих користувачів, оскільки не потрібно купувати нові камери для спостереження.

3. Налаштування та персоналізація функцій відеотрансляції в залежності від потреб користувача (використання старих телефонів як відеореєстраторів або камер спостереження за об'єктами).

Таким чином, розробка додатку відеотрансляції для мобільних пристроїв на базі Android є перспективним та соціально корисним проєктом, який дасть нове життя старим смартфонам та сприятиме розвитку технологій відеоспостереження.

Відомий на сьогодні популярний додаток Camy є додатком, який реалізує систему передачі потокового відео на різні пристрої, забезпечуючи користувачам можливість перегляду відеоконтенту в режимі реального часу [3]. Однак основна відмінність між Camy та пропонованим в роботі застосунком полягає в здатності останнього ефективно працювати з динамічними IP-адресами, усуваючи необхідність використання статичних IP-адрес завдяки інтеграції протоколу WebRTC (Web Real-Time Communication) [4, 5]. Це забезпечує спрощене налаштування, швидкий запуск та розширює можливості використання системи, оскільки користувачам не потрібно вручну конфігурувати статичні IP-адреси, що часто є складним та затратним процесом. Крім того, застосунок підтримує трансляцію з вимкненим екраном, що оптимізує енергоспоживання пристрою, особливо важливе для мобільних пристроїв, знижуючи навантаження на графічний процесор та інші компоненти.

Клієнтська програма для ПК надає можливість не лише перегляду відеопотоку, але й детального аналізу кожного кадру з налаштуванням параметрів, таких як зум, контрастність, яскравість та гамма, що покращує якість перегляду та дозволяє здійснювати точний аналіз відеоінформації. Додаток також дозволяє зберігати відео по кадрах для подальшого перегляду, що корисно для наукових досліджень, безпеки та моніторингу.

Особливою перевагою є підтримка адаптивної передачі фреймрейту та якості відео, яка автоматично знижує роздільну здатність трансляції при поганому з'єднанні, забезпечуючи стабільність передачі навіть у умовах нестабільного мережевого зв'язку. Це досягається динамічним регулюванням параметрів відеопотоку на основі оцінки стану мережі в реальному часі, що гарантує безперебійну передачу відео та мінімізує ризик втрати даних або переривання трансляції. Розроблений застосунок пропонує суттєві переваги у порівнянні з Camy, забезпечуючи гнучкість, простоту налаштування, оптимізацію енергоспоживання та високу якість відеоінформації.

Розглянуті характеристики роблять пропонований застосунок конкурентоспроможним та привабливим для користувачів, які прагнуть досягти високої якості відеотрансляцій з мінімальними витратами енергії та максимальним комфортом у використанні. Подальші дослідження можуть бути спрямовані на інтеграцію додаткових функціональних можливостей, таких як підтримка віртуальної реальності або штучного інтелекту для автоматичного аналізу відеоконтенту, що відкриває нові горизонти для розвитку систем потокового відео.

Література

1. GlobalStats statscounter. https://gs.statcounter.com/os-market-share/mobile/worldwide
2. Android documentation. https://developer.android.com/reference
3. Camy - Відео моніторинг. https://play.google.com/store/apps/details?id=cam.camy
4. WebRTC. https://webrtc.org
5. Johnston, A. B., & Burnett, D. C. (2014). WebRTC: APIs and protocols of the HTML5 real-time web (3rd ed.). Digital Codex LLC. ISBN-13: 978-0-9859788-7-7.

Додаток Б

Тези конференції

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
імені ІВАНА ПУЛЮЯ
НАУКОВЕ ТОВАРИСТВО ім. ШЕВЧЕНКА

М А Т Е Р І А Л И

XII НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

18–19 грудня 2024 року

ТЕРНОПІЛЬ
2024

УДК 001
М34

ПРОГРАМНИЙ КОМІТЕТ

Голова: Приймак Микола – професор кафедри комп'ютерних систем та мереж, д.т.н., професор.
Співголови: Марущак Павло – проректор з наукової роботи, докт. техн. наук, професор.
Баран Ігор – канд. техн. наук, доцент, декан факультету ФІС.
Науковий секретар: Надія Крива– старший викладач.

Члени: Василь Кривень – завідувач кафедри математичних методів в інженерії д.ф.-м.н., професор; Галина Осухівська – завідувач кафедри комп'ютерних систем та мереж, к.т.н., доцент; Микола Карпінський - професор кафедри кібербезпеки, д.т.н., професор; Жанна Баб'як - завідувач кафедри української та іноземних мов, к.пед. н., доцент; Ярослав Литвиненко – професор кафедри комп'ютерних наук, д.т.н., професор; Михайло Петрик - завідувач кафедри програмної іженерії, д.ф.-м.н., професор; Наталія Загородна – завідувач кафедри кібербезпеки, к.т.н., доцент.

ОРГАНІЗАЦІЙНИЙ КОМІТЕТ

Голова: Скоренький Юрій Любомирович – канд. техн. наук, доцент кафедри фізики.
Члени: доцент кафедри комп'ютерних наук, к.т.н. В. Никитюк; доцент кафедри програмної інженерії, к.т.н. Д. Михалик; доцент кафедри кібербезпеки, к.т.н. М. Стадник; доцент кафедри комп'ютерних систем та мереж, к.т.н. Є. Тиш; ст. викладач Л. Джиджора.

Матеріали XII науково-технічної конфції «Інформаційні моделі, системи та технології» Тернопільського національного технічного університету імені Івана Пулюя, (Тернопіль, 18–19 грудня 2024 р.). – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2024. – 238 с.

М34

Адреса оргкомітету: ТНТУ ім. І. Пулюя, м. Тернопіль, вул. Руська, 56, 46001,
тел. (0352) 52-41-33, факс (0352) 25-49-83.
E-mail: conffis2024@gmail.com
Редагування, оформлення та верстка: Надія Крива

СЕКЦІЇ КОНФЕРЕНЦІЇ, ЯКІ ПРЕДСТВЛЕНІ В ЗБІРНИКУ

- Математичне моделювання
- Інформаційні системи та технології, кібербезпека
- Комп'ютерні системи та мережі
- Програмна інженерія та моделювання складних розподілених систем
- Новітні фізико-технічні та освітні технології

В збірнику надруковано тези доповідей XII науково-технічної конференції «Інформаційні моделі, системи та технології» (Тернопіль, 18–19 грудня 2024 р.) за такими науковими напрямками: математичне моделювання; інформаційні системи та технології, кібербезпека; комп'ютерні системи та мережі; програмна інженерія та моделювання складних розподілених систем; новітні фізико-технічні та освітні технології.

Розрахований на науковців, викладачів та студентів вузів.

За зміст тез та дотримання норм академічної доброчесності відповідальність несе автор.

УДК 004.45
В. С. Бондаренко
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

РОЗРОБКА ДОДАТКУ ВІДЕОТРАНСЛЯЦІЇ ПІД МОБІЛЬНІ ПРИСТРОЇ НА БАЗІ ОПЕРАЦІЙНОЇ СИСТЕМИ ANDROID

UDC 004.45
V. S. Bondarenko

DEVELOPMENT OF A VIDEO BROADCASTING APPLICATION FOR MOBILE DEVICES BASED ON THE ANDROID OPERATING SYSTEM

Розроблений мобільний додаток забезпечує відеотрансляцію в реальному часі із використанням технології WebRTC. Основне призначення системи – захоплення відео через камеру мобільного пристрою та його передача іншому пристрою або серверу з мінімальними затримками. Система інтегрує функціонал WebRTC для передачі відео та аудіо, адаптивно підлаштовуючись до якості мережевого з'єднання [2].

Основні функції розробки включають захоплення та обробку медіаданих. Додаток отримує відеопотік із камери та, за потреби, аудіопотік із мікрофона[1]. Вхідні дані обробляються із застосуванням сучасних відеокодеків (наприклад, VP8, H.264) для забезпечення високої ефективності стиснення. Використання WebRTC дозволяє створювати прямі з'єднання точка-точка, використовуючи STUN/TURN сервери для пробивання NAT і забезпечення стабільного зв'язку [4].

До основних переваг розробки належать низька затримка передачі даних, що досягає 50–200 мс, адаптивність до змін у мережі, підтримка шифрування для забезпечення конфіденційності та можливість масштабування для багатокористувацького використання. Крім того, система має інтуїтивно зрозумілий інтерфейс, що спрощує її інтеграцію в різні сфери застосування.

Недоліками розробки є високе споживання обчислювальних ресурсів, залежність від якості мережі, а також складність масштабування WebRTC для великих стрімінгів. Окрім цього, не всі пристрої підтримують сучасні кодеки, що може обмежувати функціонал додатка.

Інтеграція WebRTC з платформою Android забезпечує ефективну передачу медіаданих за рахунок використання нативних бібліотек та API, оптимізованих для цієї операційної системи[3]. Це дозволяє застосунку стабільно працювати на різноманітних мобільних пристроях, надаючи високу якість відео та мінімальні затримки. Крім того, Android дозволяє тонко налаштовувати параметри WebRTC, такі як управління енергоспоживанням та оптимізація ресурсів, що підвищує надійність роботи додатку у різних мережевих умовах. Таким чином, застосунок пропонує зручний інтерфейс для реального часу відео- та аудіокомунікацій, відповідаючи сучасним вимогам мобільних технологій.

Результати тестування продемонстрували стабільну роботу системи у різних мережевих середовищах, включаючи Wi-Fi, 4G та 3G. Якість відео залишалася задовільною навіть за низької швидкості з'єднання (1–2 Мбіт/с). Система досягла високого рівня сумісності з веб-клієнтами, побудованими на основі WebRTC API. Аналіз отриманих результатів підтвердив ефективність розробки для реального часу відеотрансляцій із мінімальними затримками. У подальшому рекомендується оптимізувати використання ресурсів, розширити функціонал (наприклад, додати можливість запису відео на сервері) та вдосконалити моніторинг мережевого стану. Розробка має значний потенціал для застосування у відеоспостереженні, дистанційному навчанні та інших сферах.

Література

1. WebRTC [Електронний ресурс]. Режим доступу: https://webrtc.org.
2. Johnston, A. B., & Burnett, D. C. (2014). WebRTC: APIs and protocols of the HTML5 real-time web (3rd ed., pp. 1-10). Digital Codex LLC. ISBN-13: 978-0-9859788-7-7.
3. Android documentation. https://developer.android.com/reference.
4. Emmanuel, E. A., & Dirting, B. D. (2017). A Peer-To-Peer Architecture For Real-Time Communication Using WebRTC. *Journal of Multidisciplinary Engineering Science Studies (JMESS).*