

МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І
СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: Інформаційна комп'ютерна система для голосового зв'язку

Здобувача вищої освіти другого (магістерського) рівня,
II курсу, групи КМІз-22-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Інформаційні управляючі системи і технології в економіці»


(підпис)

Борсука Євгенія
Павловича
(прізвище, ім'я, по батькові)

Науковий керівник


(підпис)

Філоненко Михайло
Миколайович, к.ф.-м.н.,
доцент
(прізвище, ім'я, по батькові,
науковий ступінь, вчене звання)

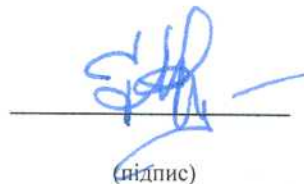
Керівник освітньо-
професійної програми


(підпис)

Горбовий Артур Юліанович,
д.т.н, професор
(прізвище, ім'я, по батькові, науковий
ступінь, вчене звання)

Допущено до захисту за рішенням кафедри
протокол від «28» грудня 2023 № 8

Завідувач кафедри


(підпис)

Філоненко Михайло
Миколайович, к.ф.-м.н.,
доцент
(прізвище, ім'я, по батькові,
науковий ступінь, вчене звання)

Ірпінь – 2024

Реєстрація № 20

30.01.2024

(дата)



(підпис відповідальної особи)

Результати перевірки

до захисту

(до захисту, на доопрацювання)

24.01.2024

(дата)



(підпис)

Результат захисту

4 (добре)

(оцінка)

09.02.2024

(дата)



(підпис)

Голова комісії:



(підпис)

В.М. Франчук

Члени комісії:



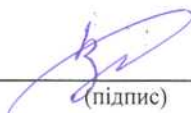
(підпис)

А.Ю. Горбовий



(підпис)

О.В. Редич



(підпис)

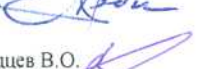
В.О. Ніжегородцев

БЛАНК ОЦІНЮВАННЯ ВИПУСКНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії, секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст випускної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	26		09.02.2024 Франчук В.М.  Горбовий А.Ю. 
2. Дотримання вимог щодо оформлення	від 0 до 10	8		Редич О.В. 
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	3		Ніжегородцев В.О.  Суховерхівка Л.П. 
4. Рецензент	від 0 до 5	4		
РАЗОМ БАЛІВ	від 0 до 50	41		

Оцінювання кваліфікаційної роботи науковим керівником - 39 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підпис)
80	B	4 (добре)	09.02.2024	Франчук В.М.  Горбовий А.Ю.  Редич О.В.  Ніжегородцев В.О.  Суховерхівка Л.П. 

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	
50-59	E	3 (задовільно)
35-49	FX	2 (незадовільно)

Факультет фінансів та цифрових технологій
Кафедра комп'ютерних та інформаційних технологій і систем
Галузь знань: 12 «Інформаційні технології»
Спеціальність: 122 «Комп'ютерні науки»
Освітня програма: «Інформаційні управляючі системи і технології в економіці»

Затверджую
Завідувач кафедри  Філоненко М.М.
« 04 » 11 2023

Завдання на кваліфікаційну роботу
здобувачу вищої освіти
Борсуку Євгенію Павловичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи : Інформаційна комп'ютерна система для голосового зв'язку
Затверджена наказом ректора від «11» жовтня 2023 № 1352
Строк здачі закінченої роботи «23» грудня 2023
Мета роботи розробити інформаційна комп'ютерна система для голосового зв'язку
Об'єкт дослідження інформаційна комп'ютерна система
Предмет дослідження – створення комп'ютерної системи голосового зв'язку між користувачами
Орієнтовна база дослідження програмні інструменти для створення комп'ютерної системи голосового зв'язку
 2. Вихідні дані до роботи комп'ютерної системи голосового зв'язку
 5. Перелік графічного матеріалу рисунки, структурні схеми
 6. Зміст кваліфікаційної роботи (перелік питань за кожним розділом)
- Розділ 1. Аналіз існуючих рішень та обґрунтування теми роботи
- 1.1 Цикл розвитку комп'ютерних систем голосового зв'язку
 - 1.2 Аналіз існуючих рішень, що формують сучасний ринок
 - 1.3 Обґрунтування теми роботи
- Розділ 2. Аналіз інструментів розробки та реалізації комп'ютерної системи
- 2.1 Обрана мова програмування
 - 2.2 Середовище розробки комп'ютерної системи
 - 2.3 Використання хмарного середовища AWS EC2
 - 2.4 Використання GitHub у проекті
- Висновки до розділу 2
- Розділ 3. Опис комп'ютерної системи та реалізації проекту
- 3.1 Архітектура комп'ютерної системи голосового зв'язку
 - 3.2 Модуль реалізації аудіозв'язку
 - 3.3 Модуль реалізації відеозв'язку
 - 3.4 Модуль інтерфейсу комп'ютерної системи
 - 3.5 Модуль ініціалізації компонентів системи
 - 3.6 Модуль реалізації серверної частини
 - 3.7 Модуль ініціалізації серверу

3.8 Модуль для роботи з мережевими пакетами

3.9 Бібліотека PyAudio

3.10 Бібліотека OpenCV

3.11. Бібліотека Tkinter

Розділ 4. Демонстрація результатів виконання дипломного проєкту

Висновки до розділу 4

Висновки

7. Календарний план виконання роботи

№ п/п	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	Фактично
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 27.10.2023	До 27.10.2023
2	I - II розділи (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	20.11.2023	20.11.2023
3	III розділ (аналіз експериментальної частини про методики досліджень)	01.12.2023	01.12.2023
4	IV розділ (проведені теоретичні і (або) експериментальні дослідження та проектні рішення)	10.12.2023	10.12.2023
5	Висновки	15.12.2023	15.12.2023
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	23.12.2023	23.12.2023
7	Отримання відгуку керівника	29.01.2024	29.01.2024
8	Отримання зовнішньої рецензії	01.02.2024	01.02.2024

8. Дата видачі завдання «04» листопада 2023

9. Керівник випускної кваліфікаційної роботи

10. Керівник освітньо-професійної програми

11. Завдання до виконання отримав здобувач вищої освіти


(підпис)

Філоненко
(прізвище, ініціал)


(підпис)

Горбовий А
(прізвище, ініціал)


(підпис)

Борсук Є.П
(прізвище, ініціал)

АНОТАЦІЯ

Інформаційна комп'ютерна система для голосового зв'язку

Мета роботи розробити інформаційна комп'ютерна система для голосового зв'язку

Об'єкт дослідження інформаційна комп'ютерна система

Предмет дослідження – створення комп'ютерної системи голосового зв'язку між користувачами

Комп'ютерна система дозволяє: здійснювати передачу голосових даних бездротовими каналами зв'язку; забезпечувати цілісність даних, що передаються. У якості середовища, що надає можливість проводити діалог між користувачами виконуючи серверну частину, було використано сервіс AWS EC2. В ході розробки:

1. Проведено аналіз методів побудови існуючих комп'ютерних систем голосового зв'язку між користувачами;
2. Сформульовано вимоги до комп'ютерної системи голосового зв'язку між користувачами згідно запропонованих особливостей;
3. Розроблено структуру комп'ютерної системи для голосового зв'язку;
4. Розроблено додаток для роботи комп'ютерної системи для голосового зв'язку;
5. Використано веб-сервіс для підтримання серверу, що надає можливість проводити діалог між користувачами використовуючи Інтернет-мережу;

Створення даного додатку спростить зв'язок користувачів через мережу Інтернет завдяки простоті використання сервісу, його прикладним особливостям та менш затратному використанню ресурсів через просту реалізацію функціональних можливостей.

Ключові слова: комп'ютерна система для голосового зв'язку, AWS EC2, Internet-мережа.

Кваліфікаційна робота містить 62 сторінки, 31 рисунок список використаних джерел з 26 найменувань.

Рік виконання роботи – 2022-2023 рр.

Рік захисту – 2024 р.

ABSTRACT

Information Computer System for Voice Communication

The aim of the work is to develop an information computer system for voice communication.

Research Object: information computer system

Research Subject: creating a computer system for voice communication between users

The computer system allows: transmitting voice data via wireless communication channels; ensuring the integrity of the transmitted data. The AWS EC2 service was used as an environment that enables dialogue between users by executing the server-side. During the development:

1. An analysis of methods for building existing computer systems for voice communication between users was conducted;
2. Requirements for the computer system for voice communication between users were formulated according to the proposed features;
3. The structure of the computer system for voice communication was developed;
4. An application for the operation of the computer system for voice communication was developed;
5. A web service was used to support the server, enabling dialogue between users using the Internet network;

Creating this application will simplify user communication over the Internet due to the simplicity of using the service, its application features, and less resource-intensive use due to the straightforward implementation of functional capabilities.

Keywords: computer system for voice communication, AWS EC2, Internet network.

The qualification work consists of 62 pages, 31 figures, a list of used sources with 26 titles.

Year of execution of the work - 2022-2023.

Year of defense - 2024.

ЗМІСТ

Вступ.....	9
Розділ 1. Аналіз існуючих рішень та обґрунтування теми роботи	10
1.1 Цикл розвитку комп'ютерних систем голосового зв'язку.....	10
1.2 Аналіз існуючих рішень, що формують сучасний ринок.....	12
1.3 Обґрунтування теми роботи.....	18
Розділ 2. Аналіз інструментів розробки та реалізації комп'ютерної системи ...	21
2.1 Обрана мова програмування.....	21
2.2 Середовище розробки комп'ютерної системи.....	23
2.3 Використання хмарного середовища AWS EC2	27
2.4 Використання GitHub у проекті	29
Висновки до розділу 2.....	31
Розділ 3. Опис комп'ютерної системи та реалізації проєкту	32
3.1 Архітектура комп'ютерної системи голосового зв'язку	32
3.2 Модуль реалізації аудіозв'язку	36
3.3 Модуль реалізації відеозв'язку	39
3.4 Модуль інтерфейсу комп'ютерної системи	44
3.5 Модуль ініціалізації компонентів системи	46
3.6 Модуль реалізації серверної частини	47
3.7 Модуль ініціалізації серверу	49
3.8 Модуль для роботи з мережевими пакетами	50
3.9 Бібліотека PyAudio	50
3.10 Бібліотека OpenCV	52
3.11. Бібліотека Tkinter	53
Висновки до розділу 3.....	55
Розділ 4. Демонстрація результатів виконання дипломного проєкту	56
Висновки до розділу 4.....	59
Висновки.....	60
Список використаних літературних джерел.....	61

ВСТУП

В останні десятиліття технологічний прогрес йде все стрімкішими та стрімкішими кроками, періодично опереджаючи людей що заставляє останніх його наздоганяти. Зараз все більше і більше користуємось месенджерами записуючі голосові повідомлення, коли немає можливості набирати текстові повідомлення.

Голосові технології дозволяють нам спілкуватися, передавати інформацію та виконувати різноманітні завдання за допомогою голосу, відкриваючи безліч можливостей для зручності, продуктивності і покращення якості життя.

Однією з найбільш очевидних переваг голосового зв'язку є його натуральність і простота використання. Ми всі вміємо спілкуватися голосом з самого дитинства, і це є одним з найприродніших способів взаємодії. Голосові технології забезпечують нам можливість спілкуватися з комп'ютерами, пристроями та іншими користувачами так само, як ми робимо це з реальними людьми, іноді навіть у більш зручний і ефективний спосіб.

Однак, наскільки корисними технології голосового зв'язку між користувачами можуть стати, залежить від їх вдосконалення та врахування питань безпеки та конфіденційності. Розвиток голосових технологій вимагає уваги до цих аспектів, тому ж кожне наступне рішення у цій сфері вдосконалює технології та надає можливість для розвитку ще більш інноваційних і перетворюючих особливостей у майбутньому.

Розділ 1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ РОБОТИ

1.1 Цикл розвитку комп'ютерних систем голосового зв'язку

Комп'ютерна система голосового зв'язку між користувачами стала можливою завдяки кільком важливим технологічним проривам та розвитку Інтернету. Ось деякі з ключових кроків у розвитку цієї теми:

1. Розвиток Інтернету та VoIP[1] стали одними із найбільш значущих проривів у розвитку технології передачі голосу за Інтернет-протоколами. Технологія VoIP дозволяє перетворювати аналогові аудіосигнали, що користувач передає завдяки динамікам прослуховування, на зразок мікрофону в мобільному пристрої або ж такого в персональному комп'ютері, на цифрові сигнали, які направляються по мережі завдяки протоколам передачі даних.

2. Створення перших програм для голосового зв'язку, які почали з'являтися ще наприкінці 1990-х і на початку 2000-х років з початком активного розвитку Інтернету, що набув популярності серед поширеного кола користувачів. Однією з перших таких систем було програмне забезпечення RASCAL[2] (Remote Audio Sound Card Application Link). Вона була розроблена у 1989 році одним ентузіастом – Браяном С. Вайлзом, що бажав створити додаток який давав би йому змогу спілкуватися із другими гравцями в онлайн-відеогрі, за якими він любляв проводити час. На той момент, технічно RASCAL був першою комп'ютерною системою зі застосуванням технології VoIP. Згодом, розробник дізнався про зусилля одного зі співзасновників Autodesk реалізувати корпоративний застосунок для голосового зв'язку між продуктами компанії Sun and Silicon Graphics, й у співпраці було створено у 1995 році один із перших комерційних продуктів для голосового зв'язку через мережу Інтернет, що отримав назву NetFone[2], який наслідував частину коду RASCAL.

3. Завдяки розвитку широкосмугового Інтернету стало можливим передавати більші обсяги даних швидше та ефективніше, без втрати частини пакетів, що суттєво покращило якість голосового зв'язку між користувачами.

Звертаючись до попереднього пункту, на момент створення перших комп'ютерних систем голосового зв'язку, об'єм передачі даних обмежувався від 32KB до 64KB, що для більшості компаній було граничним ресурсом для передачі даних між підприємствами або ж сателітами.

4. Згодом, досить вибуховою подією у сфері розвитку передачі голосових даних стала поява Skype та інших платформ-початківців, що буде обговорено далі. Вони вперше презентували зручний та зрозумілий інтерфейс для користувацького спілкування через систему голосового зв'язку завдяки мережі Інтернет. Проте час не стоїть на місці й вони продовжують розвиватись, вдосконалюючи функціонал та оптимізуючи передачу даних. З кожним роком з'являються все нові додатки, що приходять на заміну старим й пропонують більш ефективну можливість передачі даних, наприкладі Discord, що на сьогоднішній момент являється для більшості Інтернет-користувачів заміною Skype.

5. Мобільні програми голосового зв'язку. З нещодавньою появою смартфонів та стрімким розвитком мобільного Інтернету більшість компаній та платформ почали розробляти програмні продукти що орієнтуються на першочергове використання голосової системи зв'язку на мобільному пристрої. Завдяки поширеності мобільних девайсів та їх достатній потужності, стало можливим реалізовувати комп'ютерні системи також для ринку смартфонів, а враховуючи зручність та доступність у використанні, системи голосового зв'язку користуються великим попитом та постійно конкурують за інноваційні нововведення, що дає постійний поштовх для подальшого розвитку комп'ютерних систем голосового зв'язку.

Підсумовуючи попередні етапи, поява комп'ютерної системи голосового зв'язку між користувачами було результатом поєднання багатьох технологічних інновацій та тенденцій, включаючи розвиток інтернету, технології VoIP, програмного забезпечення та мобільних технологій.

1.2 Аналіз існуючих рішень, що формують сучасний ринок

Перейдемо до аналізу існуючих рішень. Комп'ютерні системи голосового зв'язку між користувачами є важливим елементом сучасних технологій комунікації. Існує багато пропозицій у цій області, які забезпечують зв'язок між користувачами за допомогою голосового інтерфейсу. Нижче наведено деякі з популярних рішень, які ми коротко проаналізуємо виділивши основні особливості та сфери застосування:

Систем Skype



Рисунок 1.1 – логотип Skype

Skype[3] є одним з найпопулярніших рішень для комп'ютерної системи голосового зв'язку. Він надає можливість здійснювати голосові дзвінки та відеодзвінки між користувачами через мережу Інтернет. Skype також підтримує конференційні дзвінки, обмін файлами та текстові повідомлення. Його особливостями були: розподіл каталогу користувачів сервісу по комп'ютерам користувачів, що надавало можливість не арендувати та масштабувати з часом значні серверні ресурси, а використовувати легкодоступні ресурси користувачів застосунку; завдяки цій технології, щоб зменшити навантаження на серверну частину, було реалізовано P2P[4] принцип, що є основою для голосового зв'язку. Фактично, він був одним із перших найпопулярніших застосунків для голосового зв'язку між користувачами, того ж його функціонал

з часом так і залишився на рівні базових потреб звичайних користувачів, яких він задовольняє. Проте з появою конкурентів додаток став втрачати популярність оскільки об'єднати різносторонню аудиторію виявилось досить не конкурентно.

Система Zoom

Zoom[5] є популярною платформою для відеоконференцій та голосового зв'язку. Він надає можливість проводити зустрічі в реальному часі з групами користувачів для відеоконференцій, нарад, дискусій, уроків, лекцій, вебінарів та неформального спілкування.



Рисунок 1.2 – логотип Zoom

Його особливістю є чітка диференціація у сторону організації групового зв'язку, що робить його зручним вибором у цій ніші. Також, досить зручним є те, що для приєднання до конференції не потребується наявність створеного аккаунту, що зменшує поріг для аудиторії у використанні цього застосунку, проте це створює певну проблему в захищеності голосового каналу, оскільки в такому разі отримавши ключ до конференції може приєднатися будь-який користувач.

Загалом, ця комп'ютерна система голосового зв'язку набула значної поширеності у період карантинних заходів під час пандемії COVID-19, оскільки люди почали працювати та вчитись дистанційно, й попит на такого роду

системи голосової організації стрімко підійнявся вгору. Завдяки простій реалізації та ефективності, ним міг опанувати кожен й це зіграло йому на користь у популяризації та поширенні серед користувачів

Система Discord

Discord[6] є платформою для спілкування геймерів, але вона також знайшла застосування як комп'ютерна система голосового зв'язку для інших сфер.



Рисунок 1.3 – логотип Discord

Він надає можливість голосового чату, текстового спілкування та обміну файлами. Серед особливостей даного застосунку можна виділити такий перелік, це: простий та легкий інтерфейс, який можна видозмінювати залежно від забагань користувача; можливість створювати групи, так названі “сервери”, як у публічному полі так і у приватному, до яких можуть приєднуватись користувачі за інтересами; можливість відео зв'язку, голосового зв'язку та демонстрації екрану в одному каналі; створення та підключення ботів за допомогою API, що підтримується розробниками, які надають різноманітні можливості маніпулювати вашим “сервером”.

Проте й варто врахувати гіркі моменти, що також присутні в даному застосунку, такі як: відсутність підтримки на більш ранніх версіях більшої частини операційних систем; часті збої при передачі файлів більших за 8 МБ

розмірів, оскільки платформа не позиціонує себе як середовище для обміну та зберігання великих файлів; відсутність функції запису відео та голосу, проте це можна компенсувати, написавши бота що виконає цю процедуру; досить назойлива реклама підписочного сервісу, оскільки застосунок безплатний, частина зручних речей, що спростила б використання сервісу, доступна за кошт.

Система Microsoft Teams

Microsoft Teams[7] є інтегрованою веб-платформою для співпраці і комунікації в команді. Вона надає голосові та відеодзвінки, обмін повідомленнями, спільну роботу над документами та інші функції для спілкування команд.

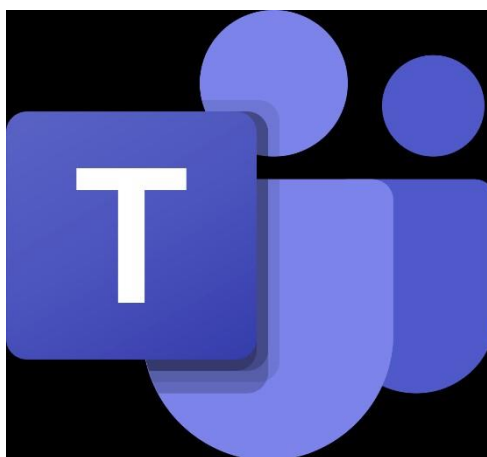


Рисунок 1.4 – логотип Microsoft Teams

У Microsoft Teams чати організовуються розмовами. Завдяки цьому всі члени команди можуть брати активну участь у спілкуванні та залишатися в курсі всіх подій та організовувати спільні заходи, визначаючи період заздалегідь.

Однією з особливостей Microsoft Teams є можливість надсилати електронні листи всьому каналу із збереженням форматування (Markdown) та вкладень. Крім того, ви зможете отримувати коментарі до листа та оповіщення. На відміну від Discord, Microsoft Teams оперує з великими

файлами та надає широку можливість ними обмінюватись та зберігати, до того ж також підтримує ботів, проте процес написання під його API менш інтуїтивно зрозумілий ніж у його конкурента. Також цікавою особливістю сервісу є те, що він нативно підтримує каталог Microsoft Office, тому ж оперувати з таким контентом у Microsoft Teams простіше.

Загалом підсумовуючи, система досить поширена у сфері бізнесу та надає можливість у масштабній організацію дистанційного або ж стаціонарного процесу в рамках одного інтерфейсу.

Система Google Meet

Google Meet[8] є інструментом для відеоконференцій та голосового зв'язку, розробленим Google. Він дозволяє здійснювати зустрічі в реальному часі з групами користувачів та має різноманітні функції, такі як екранний обмін, запис зустрічей та інші.



Рисунок 1.5 – логотип Google Meet

Конференції відбуваються у високій якості та підтримується до 250 учасників. Під час зустрічі учасники можуть спілкуватися між собою як живою, так і в чаті, проводити демонстрації екрану, використовувати інтерактивну дошку для малювання та проведення брифінгів, розміщувати учасників в особистому вікні, змінювати заднє тло та інше. Зустріч, що проводяться, захищені низкою функцій за високими стандартами захисту даних.

Особливості Google Meet: підтримка відеозустріч до 250 активних учасників, трансляція – конференція для 100 тисяч глядачів, підключення до конференцій субтитрів для іноземних учасників, планування зустрічей у календарях Google та Microsoft Outlook, проведення опитувань та запис зустрічей. Проте недоліки також присутні, серед яких: сервіс погано функціонує за слабкого інтернету. Здійснити запис відеоконференції можна лише при використанні платного тарифу.

Порівнюючи комп'ютерну систему із прямим конкурентом, Zoom, можна відмітити таку різницю як те, що перший застосунок реалізований у вигляді вебдодатку, що надає можливість отримати вихід до нього з будь-якого пристрою з підтримкою мережі Інтернет та браузеру

Система Slack

Slack[9] є платформою для комунікації та співпраці в команді. Він надає можливість голосового зв'язку, обміну повідомленнями, спільну роботу над документами та інші функції для покращення комунікації.



Рисунок 1.6 – логотип Slack

Для обговорення якихось спільних для всіх тем існують публічні канали, а також передбачені локальні групи та можливість надсилати особисті повідомлення. У месенджері можна ділитися різними файлами, він оснащений зручним пошуком, щоб можна було швидко знайти потрібні матеріали. Він

підтримує популярні хмарні сховища Dropbox, Google Диск, сервіс Google Документи, Twitter і багато інших додатків.

Особливості Slack: взаємозв'язок з іншими платформами для створення проектів, зберігання інформації, редагування документів, пересилання листів тощо; вже надіслані повідомлення редагуються, передбачені налаштування системи шаблонних та автоматичних відповідей на однотипні часті фрази; для кожного учасника діалогу встановлюється час, що відповідає його часовому поясу, що дуже важливо, коли команда велика та співробітники перебувають у різних містах.

Але навідміну від інших платформ, тут є досить суттєві недоліки: Slack у безкоштовній версії обмежує можливість пошуку повідомлень до 10 тисяч; підтримується більшість мов при написанні, проте інтерфейс локалізований лише під англійську мову; відеочат дуже повільний, та як власного немає, а інтеграція з Hangouts недостатньо добре продумана.

1.3 Обґрунтування теми роботи

1. Голосова комунікація є одним з найбільш природних та ефективних способів спілкування. Комп'ютерні системи голосового зв'язку надають можливість здійснювати голосові дзвінки, відеодзвінки та конференції через Інтернет, що робить комунікацію більш доступною та зручною для звичайних користувачів.

2. Також, завдяки швидкому розвитку технологій та зростанню віддаленої роботи, комп'ютерні системи голосового зв'язку стали невід'ємною частиною бізнес-комунікації. Вони дозволяють людям з різних куточків світу спілкуватися безпосередньо, обговорювати проекти та приймати рішення. Компанії активно використовують такі системи для підтримки комунікації своїх віддалених робітників та партнерів.

3. За останні роки швидкість та якість Інтернет-підключення значно покращилися, а мобільні пристрої стали все потужнішими та поширеними. Це

дозволяє користувачам здійснювати голосовий зв'язок за допомогою своїх смартфонів, планшетів та комп'ютерів, незалежно від їх місця перебування. Ця доступність і мобільність роблять комп'ютерні системи голосового зв'язку широко використовуваними і популярними серед користувачів, тому розвиток цього витку комунікацій є досить актуальною та популярною темою за останній десяток років і вона буде вдосконалюватись та пропонувати нові рішення задля задоволення потреб у комфорті зв'язку користувачів у різних точках світу.

4. З розвитком інноваційних технологій, таких як штучний інтелект, машинне навчання та обробка природної мови, комп'ютерні системи голосового зв'язку почали ставати більш розумними та ефективними. Такі системи можуть розпізнавати та інтерпретувати мову користувачів, надавати автоматичний переклад та аналізувати дані з голосових викликів. Ці інновації розширюють можливості комп'ютерних систем голосового зв'язку і будуть з часом інтегровані у кожен месенджер та комп'ютерну систему голосового зв'язку

5. Все актуальнішим стає подальший розвиток систем віртуальної та доповненої реальності, де голосовий зв'язок між користувачами відіграє ключову роль і надає змогу взаємодіяти не тільки невербально, а також і натуральним чином, що додає реалістичності у використанні технологій віртуальної реальності. Також, голосовий зв'язок має велику перевагу в управлінні VR/AR досвідах, що робить їх більш доступними та інтерактивними, наприкладі того, що користувач може використовувати голосові команди для маніпулювання об'єктами тощо. Не кажучи вже про ефективність, нативність та швидкість таких операцій, що є більш очевидними у використанні ніж спеціальний маніпулятор або контролер.

Завдання роботи полягає у тому, щоб написати комп'ютерну систему голосового зв'язку, яка буде задовольняти початкові потреби користувачів, мати досить доступний інтерфейс для користування програмним забезпеченням, бути достатньо ефективною та легкою у використанні з

використанням мінімальною кількістю залежностей та бібліотек, оскільки мова програмування, що була обрана для написання системи, нативно підтримується на великій кількості поширених операційних системах. Також створена комп'ютерна система має достатньо низьке використання системних ресурсів, що надає можливість запускати її на не потужному оснащенні, а з використанням стандартних для галузі комп'ютерних мереж технологій передачі аудіо та відео пакетів, надає можливість відтворювати систему на застарілому програмному забезпеченні.

Розділ 2. АНАЛІЗ ІНСТРУМЕНТІВ РОЗРОБКИ ТА РЕАЛІЗАЦІЇ КОМП'ЮТЕРНОЇ СИСТЕМИ

2.1 Обрана мова програмування

Для створення комп'ютерної системи голосового зв'язку між користувачами було обрано популярну мову програмування Python[10].



Рисунок 2.1 – логотип Python

Це високорівнева об'єкто-орієнтована мова програмування, що визначається своєю синтаксичною читабельністю та простотою. Вона була розроблена у 1991 році. Дизайн мови фокусується на читабельності коду, зменшенню його об'єму для вираження складних концепцій та конструкцій, навідмінну від інших популярних мов програмування як C++ та Java. Python є також інтерпретованою мовою програмування, що означає те, що код виконується рядок за рядком, а не компілюється заздалегідь. Це дає змогу простіше проводити процес дебагінгу та переносити на будь-яку платформу, для якої існує відповідний інтерпретатор, проте ця особливість робить інтерпретований код менш оптимізованим у часі виконання, впливає на продуктивність програми та потребує більше обсягу пам'яті для паралельної роботи інтерпретатора.

Мова програмування Python визначається динамічною типізацією типів даних. Мова автоматично визначає типи даних під час виконання програми. Це робить Python дуже гнучким в використанні, але також вимагає від програмістів бути обережними при роботі з даними різних типів.

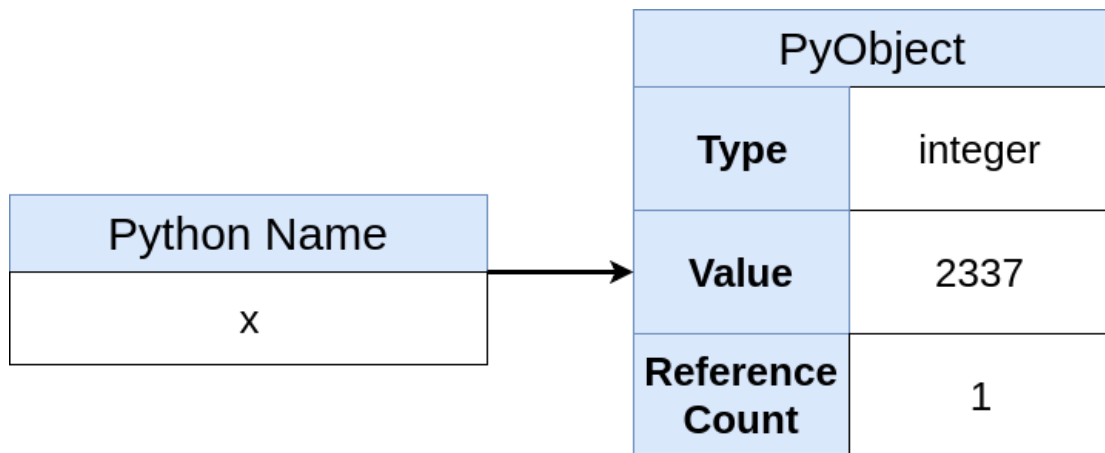


Рисунок 2.2 – схема присвоєння посилання на об'єкт який містить значення змінної

Також варто виділити особливість в управлінні пам'яттю[11] під час виконання програми. Управління ресурсами відбувається автоматично завдяки алгоритму підрахунку посилань та Garbage Collector[12]. Перший працює протягом всього виконання коду та викликається інтерпретатором в момент видалення об'єкту, на який не посилаються, а останній автоматично працює позаду сцени та виконується в основному потоці виконання програми, викликаючись з певною частотою інтерпретатором. GC в Python покликаний спростити управління пам'яттю та уникнути проблем, таких як витік пам'яті. Проте механізм GC може мати певні впливи на продуктивність програми, оскільки процес збирання сміття вимагає обробки та перебору об'єктів. Для цього він виділяє великий блок пам'яті, в якому зберігатимуться безліч невеличких об'єктів. Але зазвичай в більшості випадків ці втрати продуктивності незначні і компенсуються перевагами автоматичного керування пам'яттю, що надає GC.

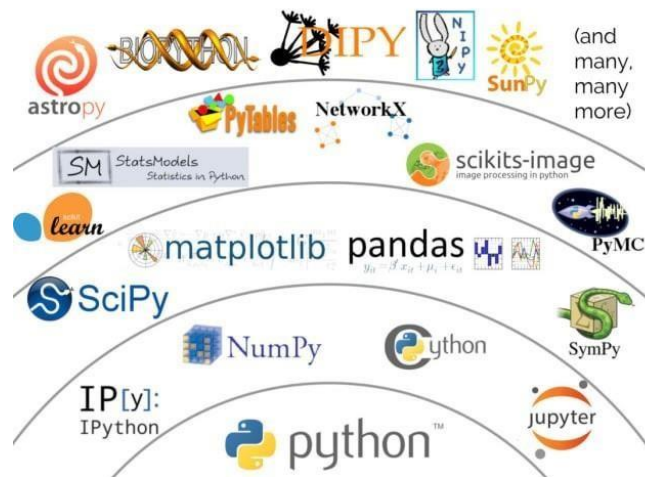


Рисунок 2.3 – одні з самих популярних бібліотек Python

Стандартна бібліотека Python[13] є однією з головних переваг мови програмування Python. Бібліотека містить вбудовані модулі, такі як `os`, `sys`, `datetime`, які надають доступ до системних функцій, роботи з файлами та часом. Крім того, бібліотека містить інші модулі, які надають стандартизовані рішення для багатьох проблем, які виникають повсякденно. Крім того, велика кількість сторонніх бібліотек та фреймворків для Python робить мову ще більш привабливою для розробки програмного забезпечення. Починаючи від бібліотек, що оперують з HTTP-запитами, закінчуючи бібліотека, що працюють із навчанням штучного інтелекту, який згодом буде виконувати поставлені виробником задачі.

2.2 Середовище розробки комп'ютерної системи

Для ефективної розробки та полегшення процесу створення, редагування, тестування та налагодження програмного коду розробники використовують такі засоби як інтегровані середовища розробки – IDE. Середовища надають розробникам набір інструментів та функцій, які допомагають збільшити продуктивність та зручність роботи з кодом.

Використовуючи інтегровані середовища розробки, програмісти можуть підтримувати більше одного проекту, працювати із множиною файлів та

керувати їх налаштуваннями. Для масштабних або однотипних розробок також передбачено створення шаблонів проектів, які надалі можна клонувати. Це

досить зручна та корисна функція, що економить велику кількість часу для виробників об'ємними проектами та зменшує ризик виникнення помилок при ініціалізації проекту.

З простого, інтегровані середовища розробки також надають можливість використовувати інструменти для автоматичної перевірки синтаксису та стилю коду, засоби підтримки тестування та налагодження. А завдяки підтримці компаній, що постійно розроблюють та підтримують середовища, й користувачам, що розвивають платформу, IDE можна розширювати, підключаючи додаткові інструменти розробки через виділену вкладку Plugins, серед яких можна визначити такі стандартні плагіни як система контролю версій та автодоповнення стандартних шаблонних дій.

Варіативність середовищ та підтримка від виробників, що постійно оновлюють графічну оболонку та стандартний інструментарій, робить інтегровані середовища розробки досить привабливими для роботи в них із бажаними мовами програмування, а завдяки спільноті користувачів, що активно розвивають плагіни які покривають ті чи інші потреби при створенні, автоматизації та редагуванню коду, досвід користування IDE лише покращується та робить їх незамінними у повсякденній розробці.

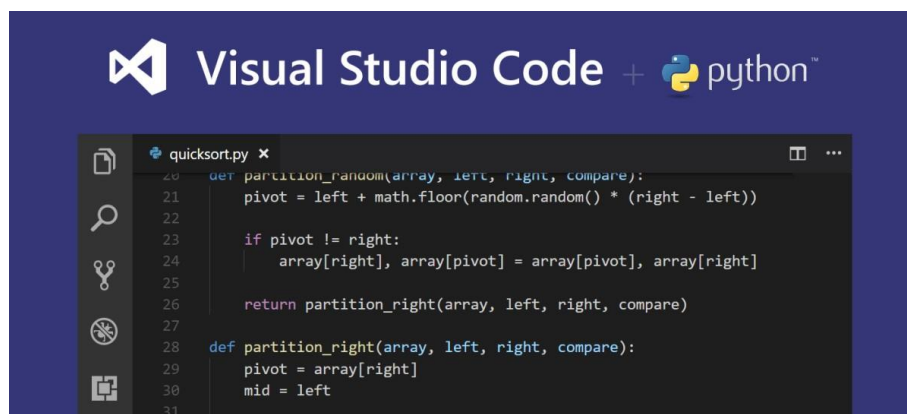


Рисунок 2.4 – інтегроване середовище розробки Visual Studio Code

Під час створення проекту я користувався таким популярним середовищем, як Visual Studio Code[14]. Він є одним з популярних інтегрованих середовищ розробки, яке надає розширені можливості для розробки різноманітних типів програмного забезпечення та дозволяє працювати із різними мовами програмування, встановивши відповідний інтерпретатор або ж компілятор та вказавши його розташування на диску. Хотів би виділити декілька переваг, що визначили розробники, які привернули мою увагу до користування цим інтегрованим середовищем розробки:

1. Кросплатформенність: VS Code доступний на Windows, macOS і Linux, що дозволяє розробникам працювати в їхніх улюблених операційних системах.

2. Розширюваність: VS Code має потужну систему розширень, яка дозволяє розробникам налаштовувати і розширювати функціональність редактора за допомогою додаткових плагінів і розширень. За допомогою цієї системи розробники можуть налаштувати своє робоче середовище з урахуванням своїх потреб і вподобань.

3. Підсвічування синтаксису та автодоповнення: VS Code надає можливість автоматичного підсвічування синтаксису для різних мов програмування і забезпечує автодоповнення коду, що допомагає розробникам писати код швидше та з меншими помилками.

4. Інтеграція з системами контролю версій: VS Code має вбудовану підтримку популярних систем контролю версій, таких як Git[15]. Це дозволяє розробникам зручно працювати зі своїми репозиторіями, відстежувати зміни, робити коміти та розв'язувати конфлікти.

5. Відладка: VS Code надає потужні інструменти для відлагодження коду, включаючи можливість встановлення точок зупинки, крокування по коду, перегляду значень змінних та використання дебаггера для різних мов програмування.

6. Інтеграція з інструментами розробки: VS Code може легко інтегруватись з іншими інструментами розробки, такими як системи збирання

проектів, сервери розробки, тестові фреймворки тощо. Це дозволяє розробникам зручно працювати з пов'язаними інструментами, не виходячи з середовища розробки.

7. Спільна робота: VS Code має функції спільної роботи, такі як можливість спільного редагування коду в реальному часі та обмін повідомленнями між розробниками, що дозволяє командам працювати над проектами разом.

8. Ефективність: Visual Studio Code відомий своєю швидкістю та ефективністю. Він працює швидко, навіть з великими проектами, і не вимагає значних ресурсів системи. Це дозволяє розробникам зосередитись на своїй роботі, не турбуючись про зависання чи повільну відгуку редактора.

9. Зручність інтерфейсу: VS Code має інтуїтивно зрозумілий та гнучкий інтерфейс. Він пропонує зручну навігацію по файлам та проектам, можливість відкривати розділені вікна для одночасного перегляду різних файлів та редакторів коду. Користувачі також можуть налаштовувати розміщення та оформлення панелей, вкладок і меню, щоб адаптувати робоче середовище до своїх власних потреб і переваг.

10. Можливість видозміни інтерфейсу: Visual Studio Code дає можливість розробникам налаштовувати та змінювати інтерфейс з допомогою розширень. Користувачі можуть встановлювати різні теми оформлення, змінювати кольори, шрифти, розміщення елементів і багато іншого. Це дозволяє створювати комфортне та

індивідуалізоване середовище розробки, що відповідає особистим вимогам та стилю роботи кожного розробника.

Ці переваги роблять Visual Studio Code потужним та універсальним інструментом для розробки програмного забезпечення різного типів та призначення.

2.3 Використання хмарного середовища AWS EC2



Рисунок 2.5 – хмарне середовище Amazon Web Service Elastic Compute Cloud

Щоб реалізувати одну з можливостей свого дипломного проекту, для хостинга серверної частини якого потребується окреме апаратне середовище, та тестування можливостей передачі даних, було обрано один із сервісів компанії Amazon. Amazon Elastic Compute Cloud (EC2)[16] є одним з ключових сервісів платформи Amazon Web Services (AWS). Він надає масштабовані обчислювальні ресурси у хмарі, що дозволяє користувачам легко запускати та керувати віртуальними серверами. Щоб охарактеризувати його, можна виділити декілька тез, такі як:

Гнучкість - AWS EC2 дозволяє вибрати необхідні обчислювальні ресурси (процесор, оперативну пам'ять, зберігання) відповідно до вимог вашого додатку або проекту. Ви можете вибрати різні типи і розміри інстансів, щоб оптимізувати продуктивність та вартість;

Масштабованість - EC2 дозволяє масштабувати ваші ресурси вгору або вниз, щоб відповідати змінним потребам вашого додатку. Ви можете легко змінювати кількість інстансів, їх розміри та конфігурацію, щоб забезпечити оптимальну продуктивність.

Автоматизація - AWS EC2 надає широкий набір інструментів для автоматизації налаштування, розгортання та керування вашими інстансами. Ви можете використовувати сервіси, такі як AWS CloudFormation, AWS Elastic

Beanstalk або Amazon EC2 Auto Scaling для спрощення процесу розгортання та керування вашими серверами.

Висока доступність - EC2 надає можливість розгортати ваші інстанси в різних регіонах та доступність зонами для забезпечення високої доступності. Ви можете використовувати різні стратегії резервного копіювання та моніторингу, щоб гарантувати безперебійну роботу вашого додатку.

До переваг використання саме хмарним сервісом Amazon Web Services Elastic Compute Cloud можна віднести такі пункти:

1. EC2 надає інтуїтивно зрозумілий інтерфейс та документацію, що допомагає швидко налаштувати та керувати віртуальними серверами.
2. Ви можете масштабувати ресурси EC2 вгору або вниз залежно від зростаючих потреб вашого бізнесу.
3. EC2 забезпечує високу продуктивність завдяки використанню потужних фізичних серверів та можливості вибору оптимальної конфігурації ресурсів.
4. AWS EC2 пропонує варіацію заходів безпеки, таких як контроль доступу, ізоляція ресурсів та шифрування даних, щоб забезпечити надійність вашого середовища.
5. AWS EC2 має широкий спектр можливостей та інтегрується з іншими сервісами AWS, що дозволяє створювати складні та розширені архітектури.
6. AWS є провідним провайдером хмарних послуг з великим досвідом та надійною інфраструктурою, що забезпечує високу доступність та безпеку ваших даних.
7. AWS EC2 надає гнучкі тарифні плани, включаючи можливість платити лише за використані ресурси, що дозволяє економити кошти.

Загалом, AWS EC2 є потужним сервісом обчислення в хмарі, який забезпечує гнучкість, масштабованість, надійність та безпеку. Він відповідає потребам різних бізнесів та додатків, дозволяючи їм легко керувати обчислювальними ресурсами у хмарі та зосередитись на розвитку своїх проєктів.

2.4 Використання GitHub у проєкті

Для зберігання розробки дипломного проєкту, було обрано платформу GitHub. Це веб-платформа для зберігання, управління та спільної роботи з репозиторіями коду за допомогою системи контролю версій Git. Вона надає зручний інтерфейс для розробників, який дозволяє взаємодіяти з репозиторіями, зберігати та відстежувати зміни у коді, спілкуватися з іншими розробниками, оглядати та рецензувати код, створювати питання та задачі.



Рисунок 2.6 – веб-платформа GitHub

GitHub[17] дозволяє розробникам зберігати весь код проєкту в репозиторіях. Репозиторій - це контейнер, який містить всі файли, теки, зміни та історію проєкту. Кожен розробник може створити свій власний репозиторій або приєднатися до існуючого. Розробники можуть працювати з кодом, створюючи гілки (branches), роблячи зміни (commits), вносячи виправлення та новий функціонал, а потім об'єднувати ці зміни з основною гілкою (branch) проєкту через процес під назвою злиття (merge).

До основних переваг використання GitHub можна віднести контроль версій, оскільки GitHub заснований на Git, це дозволяє зберігати повну історію змін коду. Отож, розробники можуть відстежувати, хто, коли і які зміни робив у коді, що надає можливість легко відновити попередні версії коду та виявляти проблеми. Також GitHub досить зручний для спільної роботи. Він надає зручні інструменти для спільної роботи над проектами. Розробники можуть створювати питання (issues) для обговорення проблем, створювати задачі (tasks) для виконання, обговорювати зміни коду, проводити рецензії та об'єднувати зміни в основну гілку (branch). У тому числі GitHub є чудовою платформою як сховище коду оскільки це централізоване місце для зберігання коду, до якого можна отримати доступ з будь-якого місця, що дає можливість розробникам працювати з кодом на різних комп'ютерах та легко ділитися ним з колегами. Й не менш важливою перевагою є спільнота розробників: GitHub є популярною платформою серед розробників, і нею користується активна спільнота. Це означає, що ви можете знайти відкриті проекти, внести свій внесок, залучитися до проектів інших розробників та отримати підтримку та зворотний зв'язок.

В цілому, GitHub допомагає розробникам зберігати, відстежувати та спільно працювати над кодом, роблячи розробку програмного забезпечення більш організованою, ефективною та сприяючи спільному розвитку проектів.

Висновки до розділу 2

У розділі було проведено аналіз інструментів розробки та реалізації програмного забезпечення.

В результаті аналізу було виділено наступні технології:

1. Мова програмування Python, завдяки своїй простоті, високому рівню абстракції та достатньому відношенню потужності та зручності використання із широким набором інструментів та бібліотек, що підтримуються спільнотою.

2. Інтегроване середовище розробки VS Code для зручної роботи з кодом, редагуванням, тестуванням та налагодженням кінцевої комп'ютерної системи. Завдяки простоті в інтерфейсі, швидкодії, можливості власноруч вибрати інтерпретатор потрібної версії, можливості налагодження середовища під власні потреби та широкий пропозиції плагінів для вдосконалення досвіду від використання IDE було звернуто увагу саме на цей продукт.

3. Хмарний сервіс Amazon Website Service Elastic Computing Cloud для хостингу серверної частини комп'ютерної системи голосового зв'язку між користувачами. Він надає масштабовані обчислювальні ресурси у хмарному середовищі, що дозволяють запустити віртуальний сервер, використовуючи досить зрозумілу документацію для налагодження серверу та його користуванням.

4. Веб-платформа GitHub для зберігання коду, контролю версіями та простим поширенням серед інших користувачів. Завдяки цій платформі можливе розташування коду в місці, з якого можна буде завантажити проект та запустити його з будь-якої машини із підтримкою необхідних вимог для його коректної роботи.

Таким чином, в результаті аналізу було визначено набір інструментів, що допоможуть у ефективній реалізації комп'ютерної системи голосового зв'язку між користувачами.

3 ОПИС КОМП'ЮТЕРНОЇ СИСТЕМИ ТА РЕАЛІЗАЦІЇ ПРОЄКТУ

3.1. Архітектура комп'ютерної системи голосового зв'язку

Діаграми та блок-схеми дозволяють нам візуалізувати архітектуру застосунків та процеси, що в них відбуваються. Вони є потужними засобами комунікації між розробниками, архітекторами та іншими зацікавленими сторонами. Використання цих інструментів допомагає зрозуміти структуру застосунку та його функціонал, а також виявляти можливі проблеми та ризики на ранніх етапах розробки.

Однією з основних переваг використання діаграм та блок-схем є їх здатність до простої та логічної візуалізації складних систем. Вони дозволяють розбити систему на компоненти та модулі, показати взаємозв'язки між ними та структуру інформаційного потоку. Це сприяє більшій зрозумілості архітектури та полегшує спілкування між учасниками проекту. Крім того, використання діаграм та блок-схем дає змогу виявити потенційні проблеми та помилки в архітектурі ще до початку розробки. Виразне представлення логічного потоку допомагає ідентифікувати можливі узагальнені помилки, а також забезпечує більшу увагу до деталей та взаємозв'язків між компонентами системи.

Застосування діаграм та блок-схем також дозволяє полегшити процес розробки та підтримки застосунків. Вони стають основою для документації та розуміння системи як для поточних, так і для майбутніх команд розробників. Ці інструменти можна використовувати для автоматичного генерації коду або сценаріїв тестування, що прискорює розробку та зменшує ймовірність помилок.

Тому ж використання такого засобу побудови архітектури комп'ютерної системи є необхідним та зручним інструментом, який допомагає краще

зрозуміти взаємозв'язок компонентів системи та наглядно продемонструвати залежність окремих модулів.

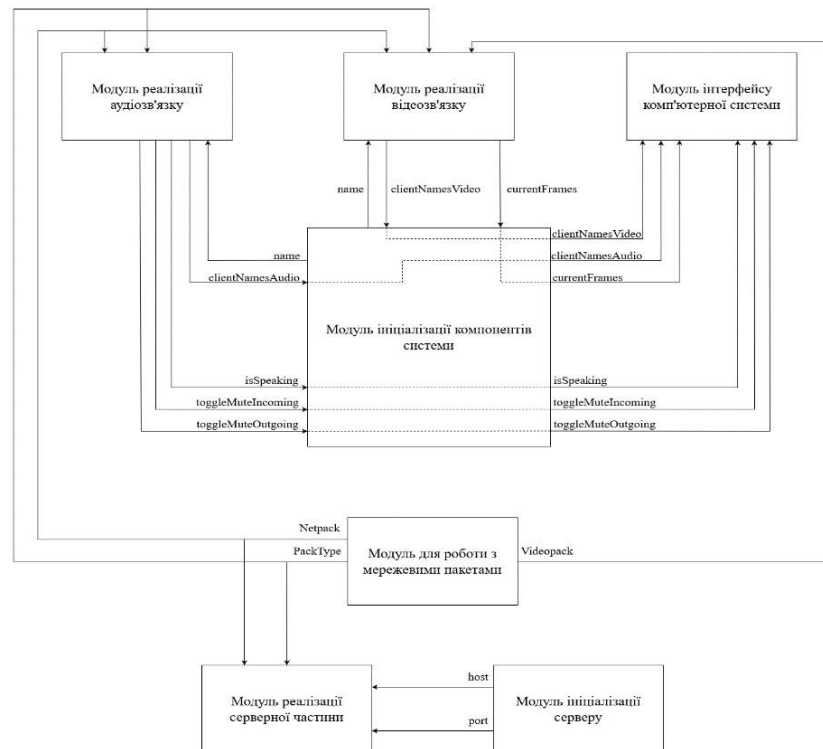


Рисунок 3.1 – архітектура застосунку «Комп'ютерна система голосового зв'язку між користувачами»

Данна комп'ютерна система складається із декількох модулів, кожен з яких реалізує окремо свій фрагмент що згодом формує зазначений програмний застосунок. Цю систему об'єднує спільна задача з обробки, передачі та відтворення мультимедійних даних (аудіо та відео даних) на основі протоколу UDP[18][19] через мережу від виділеного серверу до користувача.

Виділимо ключові компоненти даної системи голосового зв'язку:

1. Модуль ініціалізації компонентів системи (shareClient.py) – цей модуль є основною виконавчою оболонкою клієнта, яка поєднує у собі функції ініціалізації аудіо клієнта, відео клієнта та графічної оболонки системи. Усередині класу MediaClient основного модулю ініціалізуються екземпляри AudioClient та VideoClient, які потім використовуються для обміну мультимедійними даними із сервером. Крім того, цей модуль отримує введення

користувачем ім'я, що згодом передає для клієнтів, а також керує інтерфейсом користувача за допомогою передачі екземпляру GUI[20], що згодом відображає передану інформацію на екрані користувача.

2. Модуль реалізації відеозв'язку (videoClient.py) – цей модуль надає функціонал відеоклієнта. Він створює UDP-з'єднання із сервером та оброблює передачу відеоданих. Усередині класу VideoCamera кадри відео зчитуються з веб-камери користувачів, потім передаються класу VideoClient, де вони кодуються та відправляються на сервер у вигляді частин визначеного об'єму (chunks) із використанням класу Videopack для формування пакетів даних. Крім того клієнт також, окрім відправлення відеоданих, приймає відеофрагменти, декодує їх, об'єднуючи фрагменти у словнику в кадр, та відтворює їй, передаючи відеозображення на інтерфейс користувачів.

3. Модуль реалізації аудіозв'язку (audioClient.py) – цей модуль надає функціонал аудіоклієнта. Він створює UDP-з'єднання із сервером та займається обробкою, надсиланням та прийомом аудіопакетів. Усередині класу AudioClient аудіодані зчитуються із мікрофону користувача, згодом передаються методом для обробки аудіоданих у потрібному форматі, кодуються, розбиваються на частини (chunks) та надсилаються до серверу, де згодом вони будуть передані всім користувачам, окрім того, що розіслав аудіопакет.

4. Модуль інтерфейсу комп'ютерної системи (clientGUI.py) – модуль, що надає графічний інтерфейс комп'ютерній системі для користувача для зручного оперування вхідними та вихідними аудіо-відео даними. У ньому передбачено виведення зображення кожного з користувачів, що під'єднались до серверу для обміну даними, голосовий зв'язок між користувачами та реалізація функцій блокування вхідних та вихідних аудіоданих.

Модуль для роботи з мережевими пакетами (netpack.py) – модуль, що відповідає за формування пакетів із певним типом даних, будь-то користувацькі дані з передачі аудіо-відео даних, або ж пакети що відповідають за першочергове встановлення зв'язку між сервером та клієнтом, підтримкою

зв'язку між сервером та клієнтом. Також реалізований окремий клас Videopack, що наслідується від загального батьківського класу Netpack, який більш специфічно виконує обробку відеоданих з передачі їх на сервер.

6. Модуль реалізації серверної частини (mediaServer.py) – модуль, що займається обробкою вхідних пакетів даних, відповідно до типу якого, проводить певні маніпуляції з їх вмістом. Наприклад, розсилає пакети всім іншим користувачам окрім того що надіслав, або ж обробляє запити на підключення до серверу чи аналізує подальшу наявність підключення до серверу, перевіряючи стан ping-пакетів.

7. Модуль ініціалізації серверу (shareServer.py) – цей модуль задовольняє параметри запуску серверу на окремому виділеному хості, відкривши сервери для аудіо даних та відео даних. Використовується опціонально, оскільки з кожним перезапуском інстансу AWS EC2 міняє публічну IPадресу, тому ж вказувати її в окремому модулі набагато зручніше та практичніше з точки зору організації процесу.

Наразі можу ствердити, що проектування комп'ютерної системи зі строгою архітектурою за принципом модульності кожного з компонентів, що відповідає за власну реалізацію одної з важливих елементів проекту, виявляється досить зручною практикою і це досить легко можна відобразити наглядно завдяки діаграмам. Кожен з блоків дає можливість поокремо розглянути кожну з компонентів, що задіяна для формування єдиної комп'ютерної системи голосового зв'язку між користувачами, глибше зануритись у реалізацію та у разі необхідності, легше проводити правки та видозміни у застосунку.

3.2 Модуль реалізації аудіозв'язку

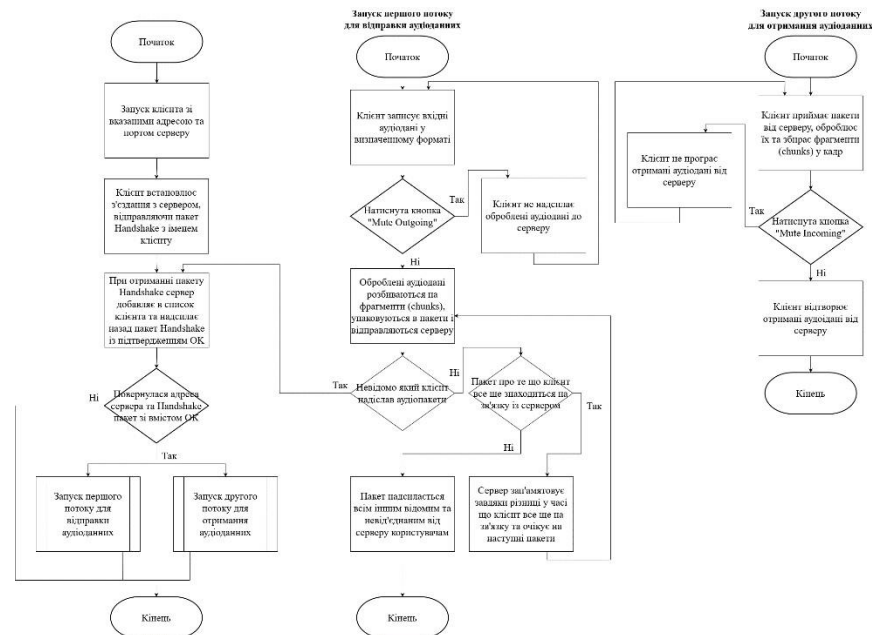


Рисунок 3.2 – алгоритм роботи модуля з передачі та прийому аудіоданих

Модуль `audioClient.py` відповідає за передачу та прийом аудіоданих між користувачами в реальному часі через використання сокетів[21] та бібліотеки `PyAudio`[22]. Розберемо його складові детальніше.

Класи та методи:

1. `getDefaultAudioSettings`:

Метод використовується для задання типових аудіо налаштувань. Зокрема, формату даних (16-бітовий `int`), кількості каналів (моно), частоти дискретизації (44100 Гц), розміру буферу (1024) та порогу шуму (500). Ці параметри використовуються для налаштування вхідного та вихідного потоків[23] аудіоданих.

2. Клас `AudioClient`:

Цей клас представляє клієнта в системі передачі голосу. Він містить методи для підключення до сервера, надсилання та отримання аудіо, створення вихідного потоку, виконання запитів до сервера та ін.

- `__init__`: Ініціалізує клієнта. Створює сокет, відкриває вхідний аудіопотік `PyAudio`, встановлює різні атрибути.

- **addOutputStream:** Створює новий вихідний потік PyAudio для користувача, ідентифікованого за допомогою унікального ідентифікатора (char).
- **connectToServer:** Підключає клієнта до сервера, відправляючи пакет рукоштовування і очікуючи підтвердження від сервера.
- **start:** Починає потоки прийому та надсилання.
- **toggleMuteIncoming:** Дозволяє вимкнути/увімкнути прийом аудіоданих.
- **toggleMuteOutgoing:** Дозволяє вимкнути/увімкнути відправку аудіоданих.
- **sendAudio:** Надсилає аудіо до сервера. Якщо аудіосигнал перевищує поріг шуму, дані відправляються. Також, спершу проходить перевірка на різницю в останньому часі відправки ping-пакету. Якщо ж він перевищує пороговий, спершу відправляється пакет для пінгування що клієнт все ще підключений до серверу.

```
def sendAudio(self):
    while self.connected:
        if time() - self.lastReceived >= START_PINGING:
            pingpack = Netpack(packType=PackType.KeepAlive, data=PINGME.encode('UTF-8'))
            self.client.sendto(pingpack.out(), self.server)

        if not self.muteOutgoing:
            data = self.streamIn.read(self.audioSettings['CHUNK'], exception_on_overflow=False)
            vol = max(array('h', data))
            if vol > self.audioSettings['THRESHOLD']:
                datapack = Netpack(packType=PackType.ClientData, data=data)
                self.client.setblocking(False)
                try:
                    self.client.sendto(datapack.out(), self.server)
                except BlockingIOError:
                    pass
                finally:
                    self.client.setblocking(True)
                    self.isSpeaking[ME] = True
            else:
                self.isSpeaking[ME] = False
        else:
            self.isSpeaking[ME] = False
```

Рисунок 3.3 – метод передачі даних на сервер

Спершу метод робить перевірку на стану того, чи під'єднаний він до серверу завдяки поверненому від методу `connectToServer` булевому значенню. У разі позитивного значення, як було зазначено раніше, проходить перевірка на різницю в часі надсилання ping-пакету задля того, щоб проінформувати сервер про те що клієнт досі підключений.

Далі проходить перевірка на те, чи не вимкнув користувач прийом даних. Якщо не вимкнув, потік прийому аудіосигналу захоплює голос, що передає в динамік користувач. У разі, якщо записані аудіодані переступають зазначений за замовченням поріг гучності, дані (поділені заздалегідь на чанки) пакуються завдяки екземпляру класа Netpack модуля netpack.py та відправляються до серверу.

- `recieveAudio`: Отримує аудіодані від сервера і відтворює їх за допомогою вихідного потоку PyAudio. Також реалізовано фрагмент прийому оберненого ping-пакету зі сторони серверу зі вмістом того, що пінгування пройшло успішно.

```
def recieveAudio(self):
    while self.connected:
        try:
            data, addr = self.client.recvfrom(self.bufferSize)
            datapack = Netpack(datapacket=data)
            self.lastReceived = time()
            self.updateIsSpeaking()

            if not self.muteIncoming:
                if datapack.PackType == PackType.ClientData:
                    char = datapack.head
                    if self.streamsOut.get(char, None) is None:
                        self.addOutputStream(char)
                        self.queryForName(char)

                    self.streamsOut[char].write(datapack.data)

                    if self.clientNames.get(char, None) is None:
                        self.queryForName(char)

                elif datapack.PackType == PackType.KeepAlive and datapack.data.decode() == PINGME:
                    outpack = Netpack(packType=PackType.KeepAlive, data=OK.encode('UTF-8'))
                    self.client.sendto(outpack.out(), self.server)

                elif datapack.PackType == PackType.NameQuery:
                    query = datapack.data.decode(encoding='UTF-8')
                    char = int(query.split(':')[0])
                    name = query.split(':')[1]
                    self.clientNames[char] = name

        except BlockingIOError:
            pass
        except timeout:
            self.connected = False
            print('lost connection to server!')
```

Рисунок 3.4 – метод зчитування отриманих даних від серверу

Як і у попередньому методі, спершу проводиться перевірка на стан того, чи під'єднаний клієнт до серверу завдяки поверненому від методу `connectToServer` булевому значенню. Якщо було повернуто `True`, метод зчитує дані у вигляді пакетів, отримані від серверу. У разі того, якщо користувач не натискав на вимикання вхідного аудіосигналу, метод аналізує вміст пакету. Пакети, що надійшли у типі `ClientData` метод зчитує, запам'ятовує користувача, що його надіслав та відтворює користувачу вміст аудіоданих.

Якщо ж тип пакету виявився KeepAlive, яким сервер перевіряє чи підключений все ще клієнт до нього, він надсилає обернену відповідь зі вмістом даних “OK”. На кінець, якщо типом пакету виявилось NameQuery, клієнт його зчитує та записує отримані імені користувачів на сервері у список до відповідного ідентифікатору.

- queryForName: Повертає ім'я клієнта від серверу за його ідентифікатором, що вказав клієнт.
- updateIsSpeaking: Оновлює інформацію про те чи розмовляє користувач через аудіоклієнт.

Можливість формувати пакети даних є критичною для виконання передачі голосу через мережу. Модуль audioClient.py виконує цю задачу, використовуючи такі сторонні бібліотеки як socket, threading, pyaudio та модуль netpack.py, що відповідає за формування пакетів.

3.3 Модуль реалізації відеозв'язку

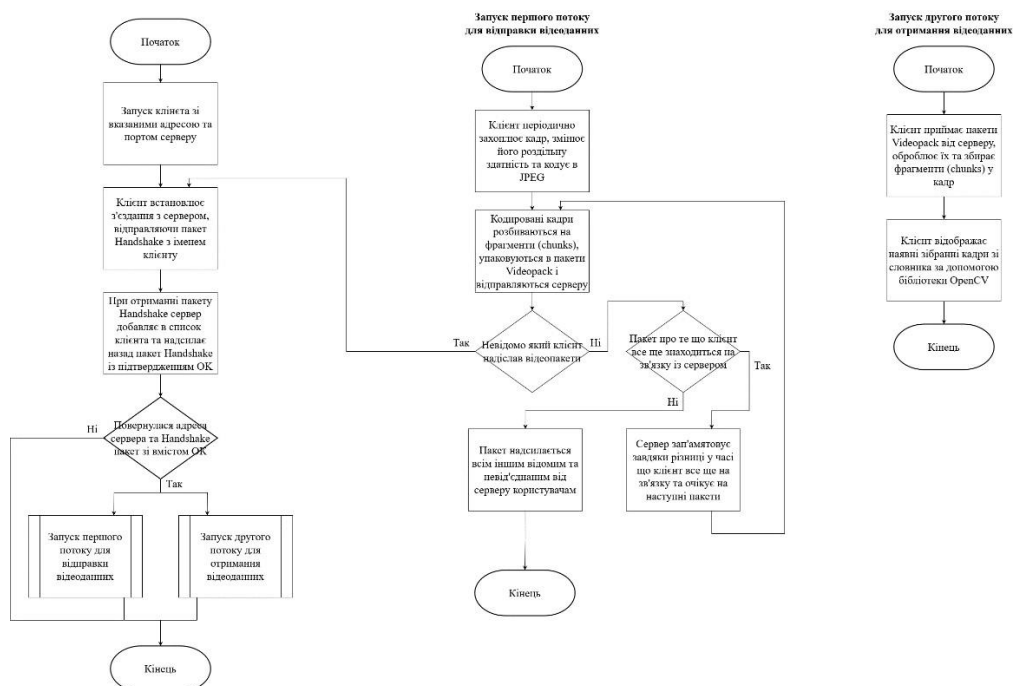


Рисунок 3.5 - алгоритм роботи модуля з передачі та прийому відеоданих

Модуль videoClient.py відповідає за передачу та прийом відеоданих між користувачами в реальному часі через використання сокетів та бібліотеки OpenCV[24]. Розберемо його складові детальніше.

Класи та методи:

1. Клас VideoCamera:

Цей клас використовує OpenCV для зчитування відео з веб-камери користувача. Він містить метод, що зчитує кадр з відео та оброблює його. • **getFrame**: Метод призначений для отримання кадру з відеопристрою і обробки його. Він змінює масштаб кадру на бажаний та кодує його у форматі JPEG.

2. Клас VideoClient:

Цей клас реалізує з'єднання із сервером, розбиває отримані від попереднього класу кадри на фрагменти, обробляє отримані від серверу фрагменти кадру, надсилає та отримує від серверу дані.

- **__init__**: Ініціалізує клієнт із заданим іменем, адресою сервера, розміром буфера та частотою кадрів, створює сокет для зв'язку із сервером.
- **connectToServer**: Підключає клієнта до сервера, відправляючи пакет рукописання і очікуючи підтвердження від сервера.
- **start**: Розпочинає потоки для надсилання та отримання відео.
- **sendFrameChunks**: розбиває кадр на фрагменти та відправляє їх на сервер у вигляді пакетів.
- **processFrameChunk**: обробляє отримані від сервера фрагменти кадрів та збирає їх у повні кадри.
-

```

def processFrameChunk(self, datapack):
    clientId = datapack.head

    newFrame = (self.chunkedFrames[clientId].get(datapack.imageId, None) is None)
    oldFrame = (not newFrame) and (time() - self.chunkedFrames[clientId][datapack.imageId].get('last', time())
    > OLD_PACKET_THRESHOLD)

    # New frame to construct
    if newFrame or oldFrame:
        self.chunkedFrames[clientId][datapack.imageId] = {}

    if datapack.index == 0:
        self.chunkedFrames[clientId][datapack.imageId]['len'] = datapack.imagelen

    # Add chunk to proper frame list
    self.chunkedFrames[clientId][datapack.imageId][datapack.index] = datapack.data
    # Update last packet received timestamp
    self.chunkedFrames[clientId][datapack.imageId]['last'] = time()

    # Check if Frame construction is complete
    imagelen = self.chunkedFrames[clientId][datapack.imageId].get('len', float('inf'))
    #currLen = number of chunks - 1 for 'last' and -1 for potentially 'len'
    currlen = len(self.chunkedFrames[clientId][datapack.imageId]) - 1
    if imagelen != float('inf'):
        currlen -= 1

    if currlen >= imagelen:
        # print('constructing...')
        if self.islater(datapack.imageId, self.chunkedFrames[clientId]['currId']):
            image = self.constructFrame( self.chunkedFrames[clientId][datapack.imageId] )
            self.currentFrames[clientId] = image
            # print('VideoClient: CurrFrames = {}'.format(len(self.currentFrames)))
            self.chunkedFrames[clientId]['currId'] = datapack.imageId
        del self.chunkedFrames[clientId][datapack.imageId]

```

Рисунок 3.6 – метод оброблення отриманих фрагментів кадру

Метод аналізує, чи є отриманий від сервера фрагмент частиною нового кадру. Якщо ідентифікатор зображення ще не зустрічався в словнику `chunkedFrames` для даного клієнта, то фрагмент є новим кадром. Наступна зміна `oldFrame` визначає, чи є отриманий фрагмент частиною старого кадру. Це відбувається, коли час останнього отриманого фрагмента для даного ідентифікатора зображення перевищує встановлений поріг, за яким як правило може слідувати передача до серверу `ping`-пакету.

Далі ми аналізуємо вміст словника, який очищаємо та якщо фрагмент має індекс 0, тобто є першим фрагментом кадру, то ми зберігаємо загальну кількість фрагментів, яка передається в даному пакеті.

Вносимо дані фрагмента у словник за відповідним індексом та оновлюємо час останнього отриманого фрагменту. Отримуємо загальну кількість фрагментів у словнику та визначаємо поточну кількість отриманих фрагментів. Надалі перевіряємо, чи отримано всі фрагменти кадру.

Згодом, робимо перевірку на змінність кадру. Якщо поточний ідентифікатор зображення пізніше від поточного ідентифікатору зображення, що міститься в словнику `chunkedFrames`, то ми збираємо новий кадр зі

словнику з отриманих фрагментів, зберігаємо повний кадр у новому словнику `currentFrames` для даного клієнта.

Підсумовуючи, метод працює з фрагментами кадрів, які прибувають, зберігає їх в словнику `chunkedFrames`, оновлює інформацію про поточний стан кадру, і коли всі фрагменти кадру отримано, він збирає кадр разом і зберігає його в словнику `currentFrames`.

- `isLater`: Метод визначає, чи є кадр з першим ідентифікатором більш «новим», ніж кадр з наступним ідентифікатором, що використовується для впорядкування кадрів відеопотоку, якщо вони приходять у різному порядку в якому були відправлені до клієнту.
- `queryForName`: Повертає ім'я клієнта від серверу за його ідентифікатором, що вказав клієнт.
- `constructFrame`: Метод що використовується для створення окремого кадру з фрагментів, отриманих з сервера. Він об'єднує фрагменти у список, а потім у байтову строку, яку згодом перетворюють в зображення. Використовуючи бібліотеку `OpenCV`, строку перетворюють в масив `NumPy`, який інтерпретується як зображення в кольоровому просторі `RGB`.
- `sendVideo`: Метод що відповідає за відправку відеопакетів до серверу.

```
def sendVideo(self):
    while self.connected:
        if time() - self.lastReceived >= START_PINGING:
            pingpack = Netpack(packType=PackType.KeepAlive, data=PINGME.encode('UTF-8'))
            self.client.sendto(pingpack.out(), self.server)
        if (time() - self.lastSent) > (1 / self.fps):
            frameBytes = self.cam.getFrame()
            self.currentFrames[ME] = cv2.imdecode(np.fromstring(frameBytes, np.uint8), cv2.IMREAD_COLOR)
            self.currentFrames[ME] = cv2.cvtColor(self.currentFrames[ME], cv2.COLOR_BGR2RGB)
            self.sendFrameChunks(frameBytes)
            self.lastSent = time()
```

Рисунок 3.7 – метод відправки відеоданих до серверу

Спершу метод перевіряє, чи підключений клієнт до серверу. У разі якщо підключення активне, порівнюється час відправки останнього пакета зі змістом `ping`. Якщо час перебільшено порівняно з останнім моментом

відправки, то серверу повторно надсилається пакет типу KeepAlive для підтвердження тривалого підключення.

Коли часу з моменту останньої відправки кадру проходить більше за сталий вираз, виконується захват кадру, що кодується та декодується задля переведення його до правильного формату. Згодом кадр розбивається на фрагменти та відправляється на сервер.

- `recieveVideo`: Метод для обробки даних з аналізом типу пакету, що надійшов від серверу.

```
def recieveVideo(self):
    while self.connected:
        try:
            data, addr = self.client.recvfrom(self.bufferSize)
            datapack = Netpack(datapacket=data)
            self.lastReceived = time()

            if datapack.PackType == PackType.ClientData:
                datapack = Videopack(datapacket=data)
                if self.chunkedFrames.get(datapack.head, None) is None:
                    self.chunkedFrames[datapack.head] = {'currId': 0}
                    self.queryForName(datapack.head)
                    self.processFrameChunk(datapack)

                if self.clientNames.get(datapack.head, None) is None:
                    self.queryForName(datapack.head)

            elif datapack.PackType == PackType.KeepAlive and datapack.data.decode() == PINGME:
                outpack = Netpack(packType=PackType.KeepAlive, data=OK.encode('UTF-8'))
                self.client.sendto(outpack.out(), self.server)

            elif datapack.PackType == PackType.NameQuery:
                query = datapack.data.decode(encoding='UTF-8')
                char = int(query.split(':')[0])
                name = query.split(':')[1]
                self.clientNames[char] = name

        except timeout:
            self.connected = False
            print('Lost connection to server!')
```

Рисунок 3.8 – метод прийому пакетів від серверу

Проводиться перевірка на активне з'єднання до серверу. Метод отримує дані від серверу, перевіряє пакет на його тип. У випадку отриманого ClientData типу пакету, метод отримує фрагменти кадру та передає його методу `processFrameChunk`, запам'ятовуючи ідентифікатор нового користувача та його ім'я, якщо воно також було невідомим при першій передачі. Якщо ж було отримано пакет типу KeepAlive, що відповідає за ping серверу, в такому випадку клієнт запаковує пакет із відповіддю “OK” та

надсилає серверу назад. На кінець, пакети типу NameQuery повертають імена користувачів, що запросив клієнт у серверу, та записують його у словник.

- `disconnect`: Метод що відповідає за від'єднання від серверу та закінчення відеозйомки з камери користувача.
- `printImg`: Метод відображення всіх поточних збережених в словнику `currentFrames` кадрів від різних клієнтів.

Модуль оброблює зображення, поділяючи його на фрагменти що згодом передаються у вигляді пакетів даних, над якими він маніпулює передаючи кадри до серверу та отримуючи і збираючи кадри інших клієнтів із серверу. Він виконує свою задачу, використовуючи такі сторонні бібліотеки як `socket`, `threading`, `opencv`, `numpy` та модуль `netpack.py`, що відповідає за формування пакетів.

3.4 Модуль інтерфейсу комп'ютерної системи

Модуль `clientGUI.py` відповідає за створення графічного інтерфейсу користувача для системи голосового зв'язку. Розглянемо його виконання детальніше.

Методи:

- `__init__`: Метод-конструктор класу `GUI`, що ініціалізує екземпляр класу `GUI`. Він приймає ряд аргументів, включаючи кореневий елемент (`root`), словники з іменами клієнтів, поточними кадрами, інформацією про те, хто зараз говорить, а також два об'єкти функції для управління звуком. Крім того, він створює кнопки для вимкнення звуку вхідного та вихідного звуку.
- `update`: Метод що займається оновленням інтерфейсу, видаляючи старий контент з `canvas`[25], оновлюючи розташування та розмір міток, а також кольори обрамлення на основі того, хто зараз говорить.
-

```

def update(self):
    self.canvas.delete('all')

    newLabels = ((len(self.labels) != len(self.currentFrames)) or
                 (self.canvas.winfo_width() != self.__lastWidth) or
                 (self.canvas.winfo_height() != self.__lastHeight))

    if newLabels:
        for lbl in self.labels:
            lbl.destroy()
        self.labels = [None for client in self.currentFrames]

    self.__lastHeight = self.canvas.winfo_height()
    self.__lastWidth = self.canvas.winfo_width()

    audioNameToNum = {v:k for k,v in self.clientNamesAudio.items()}

    for i, (clientNum, frame) in enumerate(sorted(self.currentFrames.items())):
        x,y,w,h = self.getImageDimensions(i, len(self.currentFrames))

        #print(frame.shape)

        frameImg = Image.fromarray(frame).resize((w,h), Image.ANTIALIAS)
        photoImg = ImageTk.PhotoImage(image = frameImg)
        self.__photoImages[clientNum] = photoImg
        self.canvas.create_image(x,y, image=photoImg, anchor='nw', tag='vid')

        bd = 2

        if newLabels:
            self.labels[i] = tk.Label(self.canvas, text=self.clientNamesVideo.get(clientNum, '???'), fg='white', bg=GREY2)
            self.labels[i].place(x=x+5*bd, y=y+5*bd, anchor='nw')

        name = self.clientNamesVideo.get(clientNum, None)
        audioClientNum = None
        if name is not None:
            audioClientNum = audioNameToNum.get(name, None)
            ringColor = BLACK if not self.clientSpeaking.get(audioClientNum, False) else GREEN
            self.canvas.create_rectangle(x+bd//2, y+bd//2, x+w-bd//2, y+h-bd//2, width=bd, outline=ringColor, tag='rect')

    self.root.after(self.delay, self.update)

```

Рисунок 3.9 – метод оновлення графічного інтерфейсу користувача

Метод видаляє всі елементи з “canvas”. Складові частини графічного інтерфейсу будуть створені заново при кожному оновленні. Перевіряємо, чи потрібно оновлювати мітки за поточною кількістю кадрів. Якщо мітки потребують оновлення, newLabels видаляє старі мітки та створює новий список, що відповідає кількості поточних кадрів. Також метод запам’ятовує значення ширини та висоти інтерфейсу порівняно із останнім визначенням. Далі метод створює словник, що відображає імена аудіоклієнтів та їх номери.

Згодом, метод через цикл проходить по кожному поточному кадру відео, де отримує розмір та позицію кадру, потім створює PhotoImage об’єкт з кадру, масштабує його до потрібного розміру, вставляючи в canvas. Якщо потрібно створити нові мітки, він створює нову мітку для цього клієнта. Й на кінець, залежно від того, чи говорить відповідний аудіо клієнт, метод створює підсвічення рамки відеокадру навколо з відповідним зеленим кольором

- getImageDimensions: Метод розраховує розміри відео на основі кількості відеопотоків, які потрібно відобразити, та доступного місця на екрані

Загально описуючи модуль, код створює клас GUI для відеозв'язку, де користувач може бачити відео від інших учасників, бачити, хто зараз говорить (за допомогою зеленого обрамлення), і вимкнути вхідний або вихідний звук.

3.5 Модуль ініціалізації компонентів системи

Модуль `shareClient.py` реалізує мультимедійний клієнт, який використовує аудіо та відео сервери для отримання і передачі аудіо та відео даних. Він також включає графічний інтерфейс користувача для відображення відео даних та управління налаштуваннями аудіо. Проведемо аналіз кожної з його компонент.

Методи:

`__init__`: Конструктор приймає ім'я користувача, що ввів його у термінал, яке буде представлятися на сервері.

- `setAudioServer`: Метод встановлює сервер аудіо, до якого буде підключатися цей клієнт. Він ініціалізує `AudioClient` з відповідними параметрами та зберігає словник імен аудіо клієнтів.
- `setVideoServer`: Метод встановлює сервер відео, до якого буде підключатися цей клієнт. Він ініціалізує `VideoClient` з відповідними параметрами та зберігає словник імен відео клієнтів.
- `startAudio`: Метод запускає аудіо клієнт. Він зберігає словник, який вказує, які аудіо клієнти в даний момент говорять.
- `startVideo`: Метод запускає відео клієнт. Він зберігає словник, який зберігає поточні кадри для відображення на пристрої виводу.
- `toggleMuteIncoming`: Метод перемикає заглушення вхідного аудіосигналу при зміні відповідного стану користувачем.
- `toggleMuteOutgoing`: Метод перемикає заглушення вихідного аудіосигналу при зміні відповідного стану користувачем.
- `startGui`: Метод створює GUI з налаштування аудіо та відео клієнтів.

Модуль є зручним інструментом для передачі необхідних залежностей до певних модулів, допомагає простіше ініціалізувати та запустити водночас декілька клієнтів, й слугує для передачі поточної адреси серверу.

3.6 Модуль реалізації серверної частини

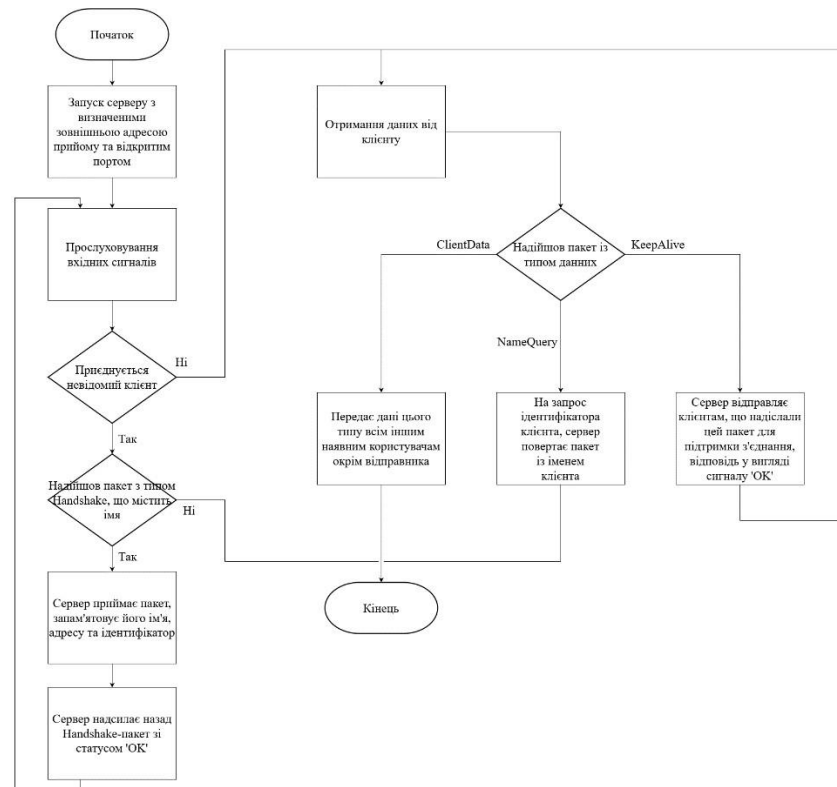


Рисунок 3.10 – алгоритм роботи серверу з прийому та передачі пакетів

Модуль `mediaServer.py`, що реалізує серверну частину комп'ютерної системи голосового зв'язку між користувачами для прийому та передачі пакетів між користувачами системи. Розглянемо його побудову та проаналізуємо його функціональність.

Методи:

- `__init__`: Конструктор класу `MediaServer`, що приймає адресу, порт та буферний розмір передачі даних, ініціалізуючи їх, створює сокет для подальшого прийому пакетів.
- `start`: Метод, що прив'язує сервер до заданої адреси хоста та розпочинає прослуховування даних у окремому потоці.

- **recieveData:** Метод безперервно отримує дані від клієнтів та включає обробку пакетів даних. Перевіряє стан з'єднання клієнтів, що знаходяться на зв'язку із сервером.

handlePacket: Метод обробки пакетів даних, що надійшли від користувача.

```
def handlePacket(self, datapack, addr):
    if self.clients.get(addr, None) is None:
        try:
            if datapack.PackType != PackType.Handshake:
                return

            name = datapack.data.decode(encoding='UTF-8')

            self.clients[addr] = name
            self.clientCharId[addr] = len(self.clients)
            self.timeOfLastMessage[addr] = time()
            self.missedPings[addr] = 0

            print('{} has connected to {} on {}'.format(name, self.tag, addr))
            datapack = Netpack(packType=PackType.Handshake, data=OK.encode(encoding='UTF-8'))
            self.server.sendto(datapack.out(), addr)
        except:
            pass
        return

    self.timeOfLastMessage[addr] = time()
    if datapack.PackType == PackType.ClientData:
        self.broadcastData(addr, datapack)
    elif datapack.PackType == PackType.KeepAlive:
        self.missedPings[addr] = 0
        if datapack.data.decode() == PINGME:
            pingpack = Netpack(packType=PackType.KeepAlive, data=OK.encode(encoding='UTF-8'))
            self.server.sendto(pingpack.out(), addr)
    elif datapack.PackType == PackType.NameQuery:
        char = int(datapack.data.decode(encoding='UTF-8'))
        addrQuery = {v:k for k,v in self.clientCharId.items()}[char]
        namepack = Netpack(packType=PackType.NameQuery,
            data=(str(char)+' '+self.clients[addrQuery]).encode(encoding='UTF-8'))
        self.server.sendto(namepack.out(), addr)
```

Рисунок 3.11 – метод аналізу отриманих пакетів

Першим чином метод обробляє нові підключення. Якщо адреса користувача невідома, а його пакет дати призначений для рукоштовування (Handshake[26] – встановки першочергового з'єднання), то метод декодує ім'я клієнта і вносить його у свій список, запам'ятовуючи також його ідентифікатор, час останнього повідомлення і кількість пропущених пінгів. Після цього він повертає клієнту назад пакет типу Handshake із відповіддю “OK”.

Якщо клієнт вже відомий, тоді метод переходить до обробки даних. Здійснюється перевірка типу пакету даних. Якщо він містить дані клієнта, то дані передаються всім іншим клієнтам підключеним до серверу окрім власне

того що надіслав пакет. У разі, якщо ж пакет містить тип KeepAlive для підтримки активності з'єднання клієнта із

сервером, кількість пропущених пінгів анулюється. Сервер відправляє назад пакет типу KeepAlive із відповіддю “OK”. На кінець, якщо клієнт бажає отримати ім'я іншого клієнта, надіславши пакет типу NameQuery, сервер шукає у своєму списку відповідне ім'я клієнта та відправляє назад його у вигляді пакету NameQuery.

- `checkConnections`: Метод перевіряє активність підключення клієнтів, відправляє пінг-пакети до неактивних клієнтів, що давно у рамках константи часу не надсилали оберненого повідомлення та відключає недоступних клієнтів.
- `disconnectClient`: Метод відключає клієнта від сервера, видаляє всю інформацію про цього зі списків сервера.
- `broadcastData`: Метод передає дані від одного клієнта до всіх інших підключених клієнтів. Використовується для передачі в методі `handlePacket`.

Серверна частина для прийому та передачі аудіо/відео даних служить для обробки та координації комунікації між різними клієнтами. Вона відповідає за обробку нових підключень до сервера, ідентифікації та ведення реєстру активних клієнтів; прийом даних від клієнтів і пересилання цих даних іншим підключеним клієнтам, дані яких містять аудіо та відео потоки; обробка запитів від клієнтів; підтримка активності за допомогою періодичного пінгування активних клієнтів; відключення неактивних від серверу.

3.7 Модуль ініціалізації серверу

Модуль `shareServer.py`, що відповідає за запуск двох екземплярів серверів для прийому аудіо та відео даних від користувачів, що встановлюють з'єднання до нього. Модуль використовується також для практичного та зручного задання актуальних даних про адресу та порти віртуального серверу, на якому хоститься серверна частина.

3.8 Модуль для роботи з мережевими пакетами

Модуль для спрощеного створення, обробки та передачі пакетів даних в мережі між користувачами. Розглянемо, з чого він складається.

Класи та методи:

1. `PackType`: Клас-енумератор, який визначає типи пакетів, що

використовуються в мережевій комунікації. Містить в собі такі типи як `ClientData`, `Handshake`, `ServerMessage`, `KeepAlive`, `NameQuery`, з якими ми познайомились раніше.

2. `Netpack`: Клас який представляє мережевий пакет даних. Він містить функції `typeToOrd` та `ordToType` які використовуються для перекладу між енемераторами типу пакету та їх числовими кодами. Містить в собі також такі методи:

- `__init__`: Конструктор що створює пакети на основі вхідних даних або існуючого пакету даних.
- `getPackType`: Статичний метод, який повертає тип пакету, використовуючи числовий код.
- `out`: Метод що перетворює пакет в байтовий масив для передачі пакету по мережі

3. `Videopack`: Клас, що наслідує `Netpack`. Створений спеціально для обробки відеоданих. Він має ті ж властивості та методи, що й батько, але також включає додаткові поля для ідентифікації відео, індексу фрагменту відео та довжини відео. Його метод `out` модифіковано з урахуванням того, щоб включати ці додаткові дані при формуванні байтового масиву.

3.9 Бібліотека `PyAudio`

`PyAudio` є однією з основних бібліотек Python для працювання з аудіоданими. Її можна використовувати для запису, відтворення і трансформації звукових даних. Це крос-платформенна бібліотека, яка працює

на багатьох різних операційних системах, включаючи Windows, MacOS та Linux.



Рисунок 3.12 – логотип PyAudio

PyAudio дозволяє взаємодіяти з системним аудіо на низькому рівні через PortAudio, бібліотеку, яка об'єднує різні аудіо інтерфейси. PyAudio надає класи для управління потоками аудіо, які можна використовувати для запису, відтворення або обох одночасно. Ці потоки можна налаштувати на конкретні параметри аудіо, такі як кількість каналів, частоту дискретизації, формат даних тощо.

У моєму випадку, PyAudio використовується для створення системи голосового зв'язку між користувачами. Клієнт-серверну систему, де користувачі можуть надсилати аудіо дані один одному через централізований сервер. Специфічно, PyAudio використовується в коді для створення потоків вхідного і вихідного аудіо. Вхідний потік використовується для запису аудіо від користувача, який потім надсилається на сервер. Вихідні потоки створюються для кожного користувача і використовуються для відтворення їх аудіоданих.

Методи PyAudio, які використовуються у проєкті, включають

`PyAudio.open` для створення потоків, `stream.read` для зчитування даних з вхідного потоку, і `stream.write` для запису даних до вихідного потоку. Всі ці методи необхідні для реалізації системи голосового зв'язку.

Також важливо зазначити, що код використовує порогове значення гучності `audioSettings['THRESHOLD']` для визначення, чи слід надсилати поточні аудіодані на сервер. Якщо гучність даного відрізка звуку перевищує

цей поріг, відрізок надсилається на сервер. Це може бути корисним для зменшення кількості передаваних даних та покращення загальної ефективності системи, а також пригнічувати недоречні та зайві звуки зі сторони користувачів системи.

3.10 Бібліотека OpenCV

OpenCV (Open Source Computer Vision) - це бібліотека програмного забезпечення, яка включає декілька сотень алгоритмів комп'ютерного зору. Вона доступна в різних мовах програмування, таких як Python, C++ і Java. Бібліотека працює на різних платформах, включаючи Windows, Linux і macOS.



Рисунок 3.13 – логотип OpenCV

OpenCV використовується для обробки зображень і відео, включаючи операції, які вимагають аналізу зображень, визначення об'єктів, відстеження об'єктів на відео та розпізнавання особливостей. Ця бібліотека використовується в різних областях, включаючи автономні автомобілі, розпізнавання обличчя, художній інтелект в іграх, розпізнавання об'єктів та багато іншого.

У коді проекту бібліотека OpenCV використовується для роботи з відео, зокрема, для зчитування відео, його обробки та кодування. Можна виділити декілька основних моментів її застосування:

1. `cv2.VideoCapture(0)`: Використовується для зчитування відео з вебкамери. Номер 0 вказує на те, що використовується перша веб-камера в системі, або ж єдина.

2. `self.video.read`: Цей метод використовується для зчитування кадрів з відео.
3. `cv2.resize`: Використовується для зміни розміру зображення.
4. `cv2.imencode`: Функція використовується для кодування зображення у форматі `.jpg`.
5. `cv2.imdecode`: Функція використовується для декодування зображення з байтового рядка.
6. `cv2.cvtColor`: Це використовується для зміни простору кольорів зображення.

Комп'ютерна система використовує OpenCV для зчитування відео з вебкамери, його обробки та передачі через мережу, маючи сервер, який приймає відеодані від клієнтів, а потім розповсюджує ці дані між іншими клієнтами. Це реалізовано з допомогою UDP протоколу, і для цього ви розбиваєте відеокадри на частини (`chunks`), які відправляються окремо.

3.11 Бібліотека Tkinter



Рисунок 3.14 – логотип Tkinter

Tkinter[27] є стандартною бібліотекою Python для створення графічного інтерфейсу користувача (GUI). Вона є обгорткою навколо Tcl/Tk - мови скриптів і багатоплатформеного набору інструментів для створення графічних інтерфейсів користувача.

Основна задача Tkinter полягає в тому, щоб допомогти програмістам створювати інтерфейси, що включають такі елементи, як вікна, кнопки, меню,

текстові поля та інші віджети. Вона дозволяє виконувати обробку подій, наприклад, реакцію на кліки миші або натискання клавіш.

У коді проекту бібліотека Tkinter використовується для створення GUI для системи голосового зв'язку. Використання Tkinter дозволяє легко створити вікно програми, кнопки для управління заглушенням вхідного і вихідного звуку, а також елементи для відображення відео користувачів.

Клас GUI визначає вікно застосунку, що включає в себе віджети Tkinter, такі як `tk.Button`, `tk.Canvas` і `tk.Frame`. Ці віджети використовуються для створення різних компонентів інтерфейсу користувача. Метод `update` використовується для постійного оновлення відображення відео на канвасі. Також у модулі застосована бібліотека PIL (Python Imaging Library) для роботи з зображеннями. Вона використовується для обробки кадрів відео, отриманих від користувачів. Виконується конвертація кадрів відео в зображення, змінюється їх розмір за допомогою метода `resize`, а потім отримане зображення використовується для створення картинки на канвасі Tkinter.

Таким чином, бібліотека Tkinter дозволяє створити вікно програми, а також визначити і контролювати віджети всередині цього вікна необхідний для базового функціоналу інтерфейсу.

Висновки до розділу 3

У розділі 3 проведено детальний опис програмного забезпечення та реалізації комп'ютерної системи голосового зв'язку. Спочатку представлено архітектуру цієї системи, де було розглянуто проектування системи та використання модульної архітектури, що значно спрощує процес розробки та масштабування.

Специфічні модулі системи розглянуті в пунктах 3.2 - 3.8. Ці модулі включають аудіозв'язок, відеозв'язок, інтерфейс системи, ініціалізацію компонентів, серверну частину, ініціалізацію серверу та роботу з мережевими пакетами.

У наступних пунктах розглянуто використання специфічних бібліотек PyAudio, OpenCV та Tkinter для реалізації аудіо, відео і графічного інтерфейсу відповідно. Ці бібліотеки дозволили нам створити високоякісну, інтерактивну і функціональну систему.

У результаті, створено повноцінну комп'ютерну систему голосового зв'язку, яка інтегрує в себе модульну архітектуру, що дозволяє легко масштабувати та додавати нові функції. Продемонстровано, як з використанням відповідних бібліотек можна створити ефективний та надійний продукт. Система готова до впровадження та подальшого вдосконалення, відповідно до потреб користувачів.

Розділ 4. ДЕМОНСТРАЦІЯ РЕЗУЛЬТАТІВ ВИКОНАННЯ ДИПЛОМНОГО ПРОЄКТУ

Зареєструвавшись на хмарній платформі від Amazon, обираємо сервіс EC2 (Elastic Compute Cloud) та переходимо до вкладки Instance. Створюємо свій віртуальний сервер, вказавши бажані параметри, зокрема додати правила включення для прийому UDP-протоколу.

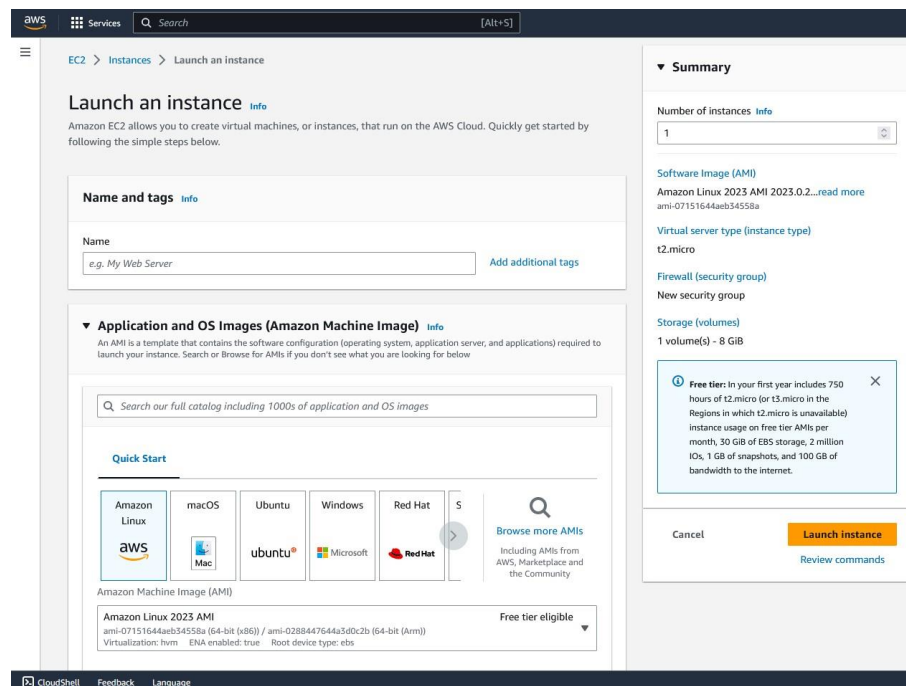


Рисунок 4.1 – створення інстансу віртуального серверу

Мій сервер працює на віртуальній машині під керуванням операційної системи Ubuntu, оскільки операційні системи на ядрі Linux за замовчуванням підтримують Python, його стандартні бібліотеки та менеджер пакетів PIP для встановлення необхідних залежностей (PyAudio, Pillow, OpenCV, Numpy).

Згодом, закінчивши вибір параметрів, створюємо пару ключів-доступу до віртуального серверу. Файл, що ми отримали, завдяки утиліті PuTTY Keygen перетворюємо у приватний ключ, який згодом використовуємо для приєднання до терміналу машини, вказавши необхідні дані з використанням згенерованого приватного ключа.

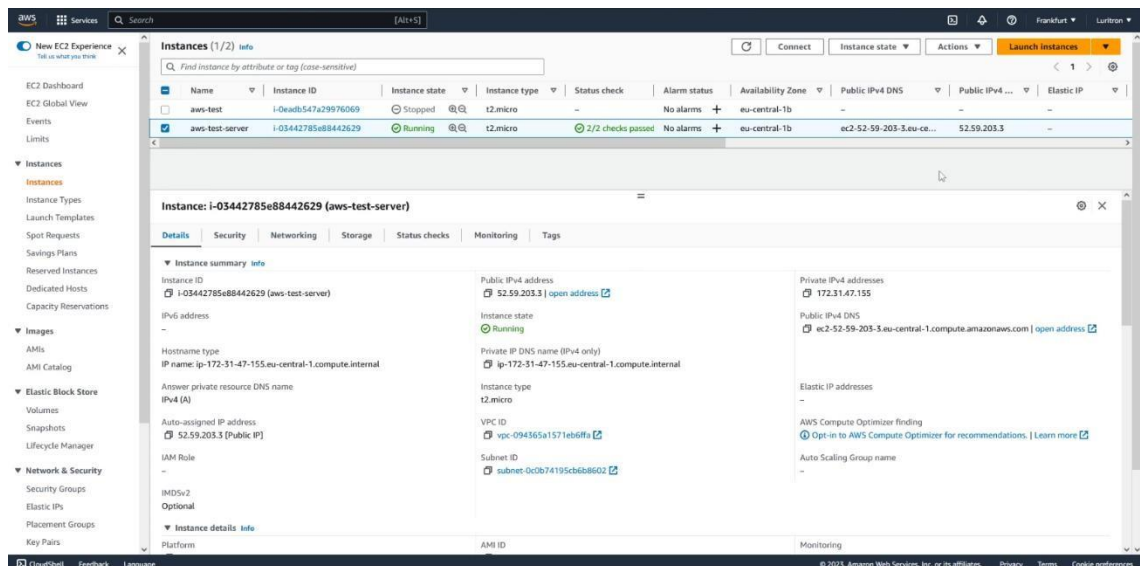


Рисунок 4.2 – основні відомості про наявний інстанс

Приєднавшись до терміналу віртуального серверу, клонуємо власний репозиторій з дипломним проектом на тему «Комп’ютерна система голосового зв’язку між користувачами».

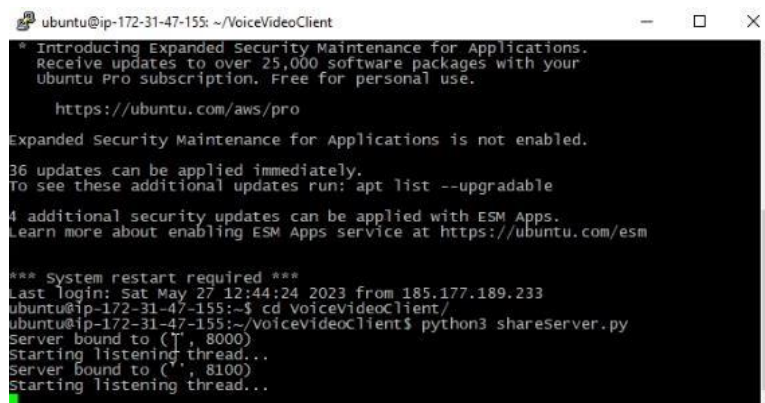


Рисунок 4.3 – запуск серверу

Запускаємо сервер, використовуючи модуль shareServer.py. Перед цим, за допомогою інструменту vim, файл було редаговано, залишивши поля IP-адрес пустими. Це означає, що доступ до серверу можна отримати як з приватної IP-адреси, так з публічної.

Тепер наш сервер відкрито для приєднання клієнтів користувачів, що бажають з ним зв’язатись.

Знаючи публічну IP-адресу нашого віртуального серверу, редагуємо поле External server у файлі shareClient.py на стороні користувача.

```

if __name__ == "__main__":
    name = input("Enter name\n")
    client = MediaClient(name=name)

    # External server AWS EC2
    client.setAudioServer('3.70.247.189', 8000)
    client.setVideoServer('3.70.247.189', 8100)

    # Local server 127.0.0.1
    # client.setAudioServer('127.0.0.1', 8000)
    # client.setVideoServer('127.0.0.1', 8100)

    while not client.audio.connected:
        client.startAudio()

    while not client.video.connected:
        client.startVideo()

    client.startGui()

```

Рисунок 4.4 – ввід IP-адреси серверу

Зберігши зміни, запускаємо клієнт та у консолі вводимо ім'я користувача. Ініціалізується інтерфейс, а у консолі з'являється повідомлення про те, що аудіо та відео клієнти користувача успішно під'єдналися до серверу. Таким чином ми встановили зв'язок із віртуальним сервером та очікуємо на другого користувача, з яким надалі можемо спілкуватись.

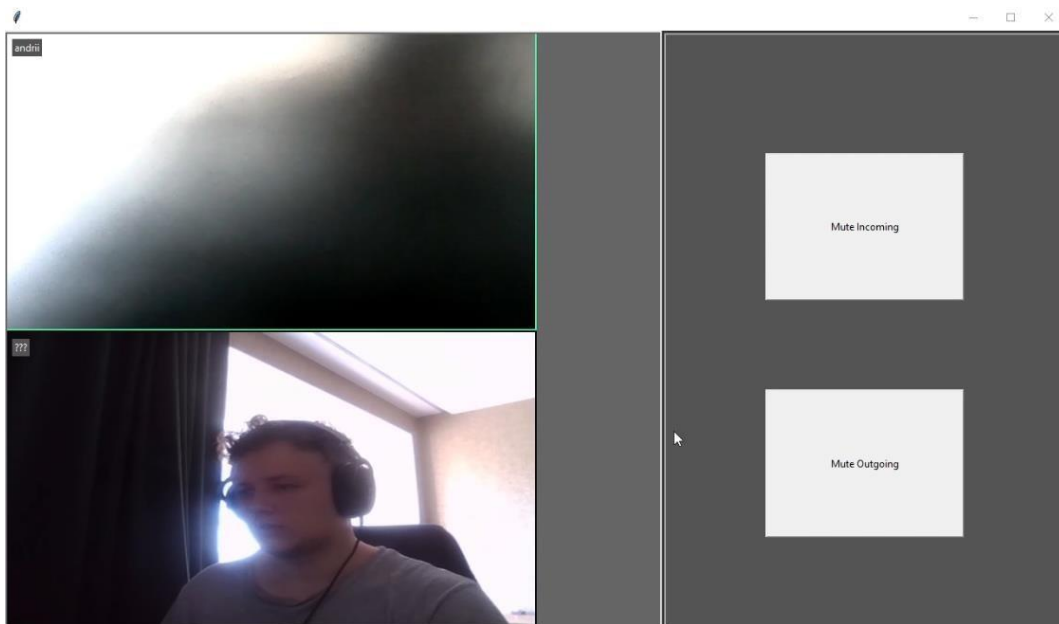


Рисунок 4.5 – демонстрація роботи інтерфейсу комп'ютерної системи

У разі того, якщо ми закінчили бесіду та хочемо вилучитись із серверу, закриваємо вікно інтерфейсу та консоль ініціалізації. Таким чином роботу системи буде завершено і користувач, що залишиться на сервері один, буде спостерігати лише останній кадр користувача, що від'єднався, не отримуючи вже від нього наступні аудіо та відео дані.

Висновки до розділу 4

У останньому 4 розділі продемонстровано принцип запуску віртуального серверу, покроково алгоритм дій якого потрібно приймати у разі того, якщо якийсь з користувачів захоче захостити бесіду з іншим користувачем/ами. Звісно, можна використовувати будь-яку іншу хмарну платформу, що надає схожі послуги, проте AWS EC2 надає зручний інструментарій для управління вхідних та вихідних протоколів, має зручний інтерфейс для налаштування інстансу, він ефективний у використанні з мінімальною затримкою даних, що пересилаються поміж клієнтів.

Також, наскільки це можливо, було відображено процес підключення користувача до створеного віртуального серверу. Користувачі змогли встановити зв'язок та почати обмінюватись даними, спілкуватися між собою при тому спостерігаючи один одного. У разі необхідності, використовуючи кнопки заглушення вхідних та вихідних аудіопакетів.

ВИСНОВКИ

У рамках цієї роботи здійснено аналіз існуючих комп'ютерних систем голосового зв'язку та висвітлено основні етапи їхнього розвитку. Цей аналіз допоміг виявити тенденції ринку, зрозуміти ключові функціональні вимоги та визначити виклики, які стоять перед сучасними системами зв'язку. Зрозумівши ці аспекти, обґрунтовано необхідність та актуальність теми дипломного проекту, обравши напрям для подальшої роботи. Запропонована система розроблена за допомогою сучасних інструментів, включаючи вибрану мову програмування, середовище розробки, хмарне середовище та інструменти контролю версій, як-то GitHub. Використання цих технологій забезпечило нам гнучкість та швидкість розробки.

Створена модульну архітектуру, що дозволила ефективно керувати різними аспектами системи, включаючи аудіо та відеозв'язок, інтерфейс користувача, ініціалізацію компонентів, серверну частину, роботу з мережевими пакетами та інше. Для реалізації використовувалися такі бібліотеки, як PyAudio, OpenCV та Tkinter тощо.

Завершальним етапом було демонстрація результатів нашого проекту. Ми успішно ініціювали віртуальний сервер на хмарній платформі AWS EC2 та демонстрували процес приєднання клієнтів до серверу.

В цілому, робота демонструє успішну реалізацію комп'ютерної системи голосового зв'язку. Розроблено функціональну, надійну та гнучку систему, яка враховує сучасні тенденції та вимоги до комунікації. Вона може бути використана як основа для подальшої роботи над розширенням функціональності та оптимізації продуктивності.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Голос через Інтернет-протокол. *Voice over IP* : URL: https://en.wikipedia.org/wiki/Voice_over_IP
2. Історія Speak Freely. *Speak Freely History* : URL: <http://www.speakfreely.org/history.html>
3. Скайп. *Skype* : URL: <https://www.skype.com/ru/>
4. Однорангове підключення. *Peer-to-peer* : URL: <https://en.wikipedia.org/wiki/Peer-to-peer> (Zoom. *Zoom* : URL: <https://zoom.us/ru> (дата звернення: 11.06.2023))
5. Discord. *Discord* : URL: <https://discord.com/>
6. Microsoft Teams. *Microsoft Teams* : URL: <https://www.microsoft.com/ukua/microsoft-teams/group-chat-software>
7. Google Meet. *Google Meet* : URL: <https://meet.google.com/>
8. Slack. *Slack* : URL: <https://slack.com/>
9. Python. *Python* : URL: <https://www.python.org/>
10. Як працює пам'ять в Python.: URL: <https://habr.com/ru/companies/domclick/articles/530804/>
11. Все, що потрібно знати про збирач сміття в Python. URL: <https://habr.com/ru/articles/417215/>
12. Стандартна бібліотека Python. *The Python Standard Library* : URL: <https://docs.python.org/3/library/index.html>
13. Visual Studio Code IDE. *Visual Studio Code IDE* : URL: <https://code.visualstudio.com/>
14. Git. *Git* : URL: <https://git-scm.com/>
15. Amazon EC2 Безпечні масштабовані обчислення для практично будь-якого навантаження.: URL: <https://aws.amazon.com/ru/ec2/>
17. GitHub. *GitHub* : URL: <https://github.com/>
18. Протокол дейтаграм користувача. *User Datagram Protocol* : URL: https://en.wikipedia.org/wiki/User_Datagram_Protocol
19. Зв'язок UDP. *UDP Communication* : URL:

<https://wiki.python.org/moin/UdpCommunication>

20. Програмування Python GUI за допомогою Tkinter. *Python GUI Programming With Tkinter* : URL: <https://realpython.com/python-gui-tkinter/>
21. socket – мережевий інтерфейс низького рівня. *socket – Low-level networking interface* : URL: <https://docs.python.org/3/library/socket.html>
22. PyAudio. *PyAudio* : URL: <https://people.csail.mit.edu/hubert/pyaudio/>
23. threading – Паралелелізм на основі потоків. *threading – Thread-based parallelism* : URL: <https://docs.python.org/uk/3/library/threading.html>
- Python. *OpenCV-Python Tutorials* : URL: https://docs.opencv.org/3.4/d6/d00/tutorial_py_root.html
24. Додавання елементів на Canvas.: URL: <https://metanit.com/python/tkinter/7.1.php>
25. TCP vs UDP. *TCP vs UDP* : URL: <https://habr.com/ru/articles/732794/>
26. tkinter – Інтерфейс Python до Tcl/Tk. *tkinter – Python interface to Tcl/Tk* : URL: <https://docs.python.org/3/library/tkinter.html>