НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

«На правах рукопису»
УДК 004.273

«До захисту допущено»
В.о. завідувача кафедри
_____ Едуард ЖАРІКОВ
«___»_____2021 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Програмне забезпечення та метод для відео-комунікації з можливостями розпізнавання мови»

Виконав (-ла):
студент (-ка) ІІ курсу, групи ІТ-04мп
Конорін Богдан Вікторович _____

Керівник:
Доцент кафедри ІПІ, к.т.н., доц.,
Фіногенов Олексій Дмитрович _____

Рецензент:
доцент кафедри ІСТ, к.т.н., доц.,
Корнага Ярослав Ігорович _____

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

Київ – 2021 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення інформаційних систем»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ Едуард ЖАРІКОВ

«___»_____2021р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Коноріну Богдану Вікторович

1. Тема дисертації «Програмне забезпечення та метод для відео-комунікації з можливостями розпізнавання мови», науковий керівник дисертації Фіногенов Олексій Дмитрович, к.т.н., доц., затверджені наказом по університету від «27» жовтня 2021 р. № 3587-с

2. Термін подання студентом дисертації «6» грудня 2021 р.

3. Об’єкт дослідження – міжмовна комунікація.

4. Предмет дослідження – програмне забезпечення відеоконференцій з підтримкою міжмовного автоматичного перекладу.

5. Перелік завдань, які потрібно розробити – аналіз існуючих рішень, формування вимог та критеріїв використання, розробка архітектури додатку, аналіз і вибір методів та технологій, розробка прототипу додатку, виконання експериментальних досліджень, розробка стартап-проєкту.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: діаграма послідовності процесу перекладу голосу учасника, архітектура системи, результат перевірки на антиплагіат.

7. Орієнтовний перелік публікацій: «Архітектурне рішення програмного забезпечення для відео-комунікації з можливостями розпізнавання мови».

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Графічний	*доц. Фіногенов О.Д.*		

9.Дата видачі завдання «1» вересня 2020 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання	Примітка
1	Видача завдання	01.09.2020	
2	Аналіз існуючих рішень	01.09.20-22.09.20	
3	Формування вимог і сценаріїв використання	23.09.20-06.10.20	
4	Розробка архітектури додатку	07.10.20-13.03.21	
5	Аналіз і вибір методів та технологій	14.03.21-17.06.21	
6	Розробка прототипу додатку	01.09.21-03.11.21	
7	Виконання експериментальних досліджень	04.11.21-08.11.21	
8	Розробка стартап-проєкту	08.11.21-14.11.21	
9	Оформлення пояснювальної записки	15.11.21-21.11.21	
10	Подання дисертації на попередній захист	22.11.21	
11	Подання дисертації на захист	6.12.2021	

Студент Богдан КОНОРІН

Науковий керівник Олексій ФІНОГЕНОВ

РЕФЕРАТ

Розмір пояснювальної записки – 99 аркуші, містить 20 ілюстрацій, 26 таблиць, 4 додатки і 26 посилань на джерела.

Актуальність теми. У роботі розглянуто проблему міжмовної комунікації. Виявлено потребу в засобах комунікації у реальному часі без володіння мовою співрозмовника. Розглядається задача побудови програмного продукту для відеоконференцій з можливостями розпізнаванням мови та подальшого перекладу на іноземну мову. Дослідження обумовлено необхідністю покращення користувацького досвіду при комунікації з іноземцями.

Мета дослідження. Розробка програмного забезпечення із забезпеченням методу міжмовної комунікації у відеоконференціях.

Об’єктом дослідження є міжмовна комунікація.

Предметом дослідження є архітектура програмного забезпечення відеоконференцій з підтримкою міжмовного автоматичного перекладу.

Для реалізації поставленої мети **сформульовані наступні завдання**:

— аналіз існуючих рішень;

— формулювання вимог і сценаріїв використання;

— розробка архітектури додатку;

— аналіз і вибір методів та технологій;

— розробка прототипу додатку;

— оцінка ефективності запропонованого рішення.

Наукова новизна результатів магістерської дисертації полягає в тому, що набуло подальшого розвитку архітектурне рішення програмного забезпечення для організації комунікації з розпізнаванням мови, що на відміну від аналогів, забезпечує можливості міжмовного спілкування, що значно розширює сферу застосування.

Практичне значення отриманих результатів полягає в тому, що була розроблена архітектура організації відеоконференцій із можливістю автоматичного перекладу мови співрозмовника.

Зв'язок з науковими програмами, планами, темами. Робота виконувалась на кафедрі інформатики та програмної інженерії Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського" в рамках теми «Математичні моделі та технології в СППР». Державний реєстраційний номер 0117U000914.

Апробація. Наукові положення дисертації пройшли апробацію на Всеукраїнській науково-практичній конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології», 22.11-26.11.2021, Київ, Україна.

Публікації. Наукові положення дисертації опубліковані в:

Конорін Б.В., Фіногенов О.Д. Архітектурне рішення програмного забезпечення для відео-комунікації з можливостями розпізнавання мови // Всеукраїнській науково-практичній конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології», 28.11-30.11.2021, Київ, Україна. – Київ, 2021. – С. 144-148.

Ключові слова: АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, КОМУНІКАЦІЯ, ВІДЕОКОНФЕРЕНЦІЯ, СППР, РОЗПІЗНАВАННЯ МОВИ, ПЕРЕКЛАД АУДІО-ЗАПИСУ, WEBRTC.

ABSTRACT

Explanatory note size – 99 pages, contains 20 illustrations, 26 tables, 4 annexes and 26 source references.

Topicality. The issue of interlingual communication has been discussed in the work. The need for means of communication in real time without knowledge of the language of the interlocutor is revealed. The problem of building a software product for video conferencing with the ability to recognize the language and to translate it further into a foreign language is considered. The study is driven by the need to improve the user experience when communicating with foreigners.

The aim of the study. The aim is to develop software which would provide interlanguage communication in video conferencing.

The object of the research is interlingual communication.

The subject of the research is video conferencing software that supports interlingual automatic translation.

The **following tasks** have been formulated to achieve the set aims:

— analysis of existing solutions;

— formulation of requirements and usage scenarios;

— application architecture development;

— analysis and selection of methods and technologies;

— development of the application prototype;

— evaluation of the effectiveness of the proposed solution.

The scientific novelty of the results of the master's dissertation is that the architectural solution of the software for the organization of communication with language recognition, which, unlike analogues, provides opportunities for interlingual communication and, thus, significantly expands the scope of application, has been further developed.

The practical value of the obtained results is that the architecture for organizing video conferencing with the ability to automatically translate the interlocutor's language has been developed.

Relationship with working with scientific programs, plans, topics. The work was performed at the Department of Informatics and Software Engineering of the National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute". State registration number - 0117U000914.

Approbation. The scientific provisions of the dissertation were tested at the All-Ukrainian scientific-practical conference of young scientists and students "Software Engineering and Advanced Information Technologies", 22.11-26.11.2021, Kyiv, Ukraine.

Publications. Scientific provisions of the dissertation are published in:

Konorin B.V., Finohenov O.D. The architectural solution for video communication software with speech recognition capabilities // All-Ukrainian scientific-practical conference of young scientists and students "Software Engineering and Advanced Information Technologies", 28.11-30.11.2021, Kyiv, Ukraine.

Keywords: SOFTWARE ARCHITECTURE, COMMUNICATION, VIDEO CONFERENCE, DSS, LANGUAGE RECOGNITION, TRANSLATION OF AUDIO RECORDING, WEBRTC.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ACID — Atomicity, Consistency, Isolation, Durability. Акронім, котрий описує вимоги до транзакціонної системи.

DevOps — методологія активної взаємодії спеціалістів з розробки з фахівцями з інформаційно-технологічного обслуговування та взаємна інтеграція їх робочих процесів один в одного для забезпечення якості продукту.

HTTP — Hyper Text Transfer Protocol. Протокол програмного рівня для передачі даних для розподілених систем.

JSON — JavaScript Object Notation. Текстовий формат обміну даних.

SSE — Server Side Event. Технологія відправи повідомлень від сервера до веб-браузера.

TCP — Transmission Control Protocol. Протокол передачі даних.

TSL — Transport Layer Security. Криптографічний протокол, котрий забезпечує захист передачі даних між вузлами в мережі Інтернет.

UDP — User Datagram Protocol. Протокол передачі даних, котрий толерантний до втрати пакетів даних.

URL — Uniform Resource Locator. Уніфікована адреса електронних ресурсів.

VoIP — Voice Over IP. Технологія передачі медіа-даних у реальному часі за допомогою сімейства протоколів TCP/IP.

WS — WebSocket. Протокол, котрий відповідає за взаємодію між браузером та сервером у реальному часі.

ВСТУП

Коронавірусна пандемія, що триває більше 2 років, внесла корективи на функціонування основних принципів суспільства. Незважаючи на всі негативні наслідки, з якими нам довелося зіткнутися, та незліченні карантинні заходи, що буквально паралізували будь-які процеси по всьому світу, це відкрило простір для усунення цих меж шляхом диджіталізації та більшої глобалізації Інтернет простору. Наприклад, однією з головних проблем, що виникла у будь-якої компанії та організації, що мусила продовжити своє існування за карантинних умов, це проблема комунікації. Більше не було можливості збиратися в офіси та влаштовувати звичні брифінги та наради. Саме в цей момент багатьом довелося дізнатися про ресурси та застосунки, що дозволять сприяти вирішенню цієї проблеми шляхом онлайн аудіо- та відеоконференцій. На щастя, даний вид комунікації не обмежує нас у часово-територіальному просторі, навпаки сприяє спілкуванню людей навіть на різних континентах. Але навіть на цьому етапі можуть виникнути проблеми з лінгвістичної точки зору, а саме мовний бар‘єр.

Існуючі програми для вирішення цілі вхідних та вихідних групових дзвінків, звичайно, дозволяють об’єднати абонентів, а саме дати платформу та засіб для подальшої комунікації. За потреби можна зробити груповий відеодзвінок, поділитися зображенням власного екрану, чат для всіх учасників бесіди, тощо. Якщо переглянути усі варіанти поліпшення процесу спілкування, які пропонує кожен з виробників програмного забезпечення, наразі, на жаль, не існує спеціалізованих систем, метою яких буде саме поліпшення питання мовного бар’єру, а саме розпізнавання тексту мови оригіналу та озвучення вихідного тексту мовою перекладу, що буде здійснено машинним шляхом. Деякі додатки пропонують усунення даного запиту шляхом розпізнавання тексту та формування субтитрів, які будуть відображені на екрані, але дана система має багато обмежень (невелика кількість мов, що можуть розпізнаватися, неточний переклад, тощо).

Метою даної роботи стало виявлення та обґрунтування архітектурних рішень та методів для побудови системи відеокомунікації зі спеціалізацією на

зчитування мови під час сеансу для подальшого перекладу на іноземну мову з можливістю подальшого озвучення або збереження у цифровому вигляді.

Завданням роботи є проаналізувати необхідні складові систем даного типу, побудувати архітектуру при котрій кожний компонент зможе виконувати поставлені функціональні та нефункціональні вимоги, проаналізувати та порівняти технології та бібліотеки кожного програмного елементу на сумісність та потенціальний розвиток. До таких елементів системи можна віднести: протоколи передачі відеосигналу, бібліотеки та сервіси для зчитування мови, перекладу та озвучування, сервіси роботи з аудіотреками, сховища даних та інфраструктурні елементи. Також, складність поставленої задачі полягає у тому, що необхідно враховувати різноманіття Web-клієнтів, а також стандартів, котрі вони підтримують, та версії протоколів по роботі з аудіо та відео потоками.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1 Загальні положення

Програмне забезпечення для відеоконференцій дає змогу спілкуватися в режимі онлайн для проведення аудіозустрічей, відеозустрічей та семінарів за допомогою вбудованих функцій, таких як чат, спільний доступ до екрану та запис. Це програмне забезпечення може бути неймовірно гнучким інструментом у програмній екосистемі бізнесу. Рішення для відеоконференцій усувають необхідність особистого відвідування, додаючи зручності до щоденного розкладу для всіх залучених осіб, покращуючи відносини з клієнтами та забезпечуючи відкрите і послідовне спілкування між командами.

Ці рішення можна використовувати для внутрішніх реєстрацій, конференц-дзвінків, зовнішніх зустрічей та презентацій. Багато інструментів для відеоконференцій також пропонують додаткові функції за межами самих відеоконференцій, забезпечуючи обмін файлами та функції миттєвого спілкування, які підтримують співпрацю в команді. Деякі системи відеоконференцій пропонують інтеграцію з програмним забезпеченням для автоматизації маркетингу та програмним забезпеченням CRM для синхронізації критично важливих бізнес-даних у відповідних конференціях і дозволяють впорядковувати подальші комунікації та оновлювати облікові записи контактів.

Крім простих можливостей відеодзвінків, багато продуктів для відеоконференцій мають широкий набор функцій, які підтримують співпрацю та спілкування на кількох фронтах. Це може включати можливість простого аудіодзвінка без відео, дошки та спільного використання екрана, а також інструменти запису дзвінків. Розуміння того, які з цих функцій можуть знадобитися компанії за межами відеоконференцій, є ключем до пошуку найкращого продукту.

Нижче наведено деякі основні функції програмного забезпечення для відеоконференцій, які дозволяють користувачам спілкуватися та співпрацювати в реальному часі:

— відеодзвінки (крім зустрічей віч-на-віч, високоякісні відеодзвінки - це наступне найкраще рішення для особистих зустрічей або зустрічей команди. Відеодзвінки не тільки покращують співпрацю, але й допомагають командам відчувати себе більш пов’язаними, коли вони не можуть зустрітися особисто);

— аудіодзвінки (багато інструментів для відеоконференцій пропонують можливість відвідувати відеоконференцію лише через аудіо, приєднавшись за допомогою номера для входу. Ці дзвінки схожі на типовий телефонний дзвінок або телефонну систему та використовують програмне забезпечення VoIP. Так само більшість продуктів дають учасникам можливість приєднатися до конференц-дзвінка лише з аудіо або відео та аудіо);

— запис (продукти з цією функцією дозволять користувачам записувати відео- або аудіоконференції, щоб його можна було пізніше переглянути. Деякі продукти навіть розміщують записане відео на платформі з можливістю завантажити його, щоб поділитися в організації або за її межами);

— спільний доступ до екрана (функція дозволяє учасникам ділитися своїми екранами разом із каналом вебкамери або замість нього. Програмне забезпечення для спільного використання екрана — це чудовий інструмент для спільної роботи, особливо для віддалених або гібридних команд, які беруть участь у частих онлайн-зустрічах);

— спільний доступ до документів (функції спільного доступу до документів іноді вбудовуються у функцію текстового чату, але це не є жорстким правилом. Деякі програми для відеоконференцій пропонують простий обмін документами без текстового чату);

— планування (можливість планувати зустрічі в програмі надається з деякими інструментами для відеоконференцій. Інші будуть інтегровані із зовнішнім програмним забезпеченням для планування або календаря);

— текстовий чат (деякі інструменти для відеоконференцій забезпечують текстовий чат у реальному часі, який учасники можуть використовувати разом із аудіо або замість нього. Ці текстові чати можна записувати та посилатися на них

пізніше. Деякі продукти для відеоконференцій дозволять обмінюватися миттєвими повідомленнями за межами відео зустрічей);

— презентації (хоча деякі інструменти для відеоконференцій дозволять розміщувати презентації через інтеграцію із зовнішнім програмним забезпеченням для презентацій, інші дозволять користувачам створювати та представляти слайд-шоу в програмі);

— субтитри (деякі продукти для відеоконференцій пропонують субтитри та закриті субтитри, що не тільки включає, але й дозволяє глобальним командам зменшити проблему мовних бар'єрів або культурних відмінностей).

Розробка таких систем потребує високу кваліфікацію кадрів для розробки власних протоколів та роботи з аудіо- та відеопотоками. В більшості додатків для відеокомунікації присутні такі складнощі по роботі як:

— якість дзвінків (однією з найпоширеніших проблем із програмним забезпеченням для відеоконференцій є якість дзвінків. Хоча саме програмне забезпечення може бути надійним, якість самого відео чи аудіо дуже залежить від швидкості інтернету залучених користувачів. Важливо, щоб користувачі мали доступ до надійного інтернету під час використання програмного забезпечення для відеоконференцій, щоб запобігти потенційним розчаруванням);

— проблеми безпеки (як і будь-яке програмне забезпечення, що використовується для внутрішнього та конфіденційного спілкування, покупці повинні оцінити протокол безпеки продукту. Хоча в минулому інструменти відеоконференцій були предметом порушень безпеки, постачальники вжили заходів для посилення безпеки зустрічей, наприклад, вимагали введення паролів для входу до наради, запроваджували користування кімнатами очікування та посилювали адміністративний контроль).

Так як, у даній роботі розглядається проблема розробки продукту для відеокомунікації зі спеціалізацією по роботі з мовою, то також потрібно розглядати такі функції як:

— переклад субтитрів (субтитри, найчастіше, є марними у випадку, коли вони не адаптовані до мови співбесідника, тому важливо інтегрувати онлайн

перекладач, котрий близько до реального часу зможе транслювати текст на потрібну мову);

— озвучування субтитрів (дуже зручним доповненням до перекладених субтитрів є їх озвучування. Це дозволить підтримувати комунікацію на потрібному рівні у ситуації поганого зв’язку зі сторони співбесідника, або для комфортнішого способу донесення інформації до слухача;

— робота с тембром голосу людини (додатковим функціоналом до озвучування субтитрів може слугувати гнучкість у роботі з голосом. Наприклад, голос озвучування субтитрів може бути жіночим, якщо за зчитаним тембром з аудіоканалу буде зрозуміло, що розмовляє жінка, так і навпаки – для чоловіка. Такий функціонал, також зменшить когнітивне навантаження на слухачів;

— кількість підтримуваних мов (не дивлячись на те, що у світі стандартом вважається знання англійської мови, бажано розширити цей список та надати змогу вибирати його, або імплементувати аутопідбір мови перекладу).

Окремо від функціональних вимог, потрібно розглядати нефункціональні вимоги до системи. Так як система створена для роботи з мовою та перекладом, то потрібно ураховувати також елементи правил синхронного перекладу.

До таких вимог потрібно розглядати:

— необхідність встановлення на персональний пристрій (більшість сучасних додатків для відеокомунікації потребують встановлення на персональний комп’ютер або телефон, так як це забезпечує гарну роботу з аудіо- та відеоканалами, що у свою чергу дає гарний звук, надійність та високе розширення картинки).

— швидкість затримки до зчитування (повноцінно синхронний переклад у реальному часі не є можливим, так як потрібно розуміти межу, котра відділяє частину сказаних слів, для подальшого перекладу. Усі аудіосистеми з розпізнаванням голосу та виконування команд працюють саме так. Потрібно визначити та забезпечити цю межу для підвищення якості перекладу);

— швидкість перекладу тексту (за стандартом для перекладачів-синхроністів, вони повинні перекладати з затримкою не більше, ніж 7 секунд. Саме такого рівня потрібно дотримуватись у додатках, що розробляються);

— підтримка кількості учасників (прямий наслідок масштабування системи та коректної конкурентної багатопотокової роботи системи. Чим більшу кількість учасників може підтримувати система, тим більш вона є привабливою для кінцевого користувача);

— ціна масштабування (робота з такими складними процесами потребує багато ресурсів, для забезпечення високих показників функціональних та нефункціональних вимог. Тому потрібно ретельно аналізувати свій бюджет та ціни на споживані сервіси);

— особливості способу трансляції (розширювання функціоналу прямопропорційно залежить від вибраних протоколів та стандартів по роботі з медіапотоками та сумісність їх з існуючими сервісами, тому також важливо ураховувати і цей фактор).

1.2 Програмне забезпечення Skype

Одним з найрозповсюдженіших додатків для відеоконференцій є Skype. Skype це peer-to-peer VoIP клієнт [1], котрий дозволяє голосові та текстові повідомлення з іншими користувачами додатку (рис 1.1). Він зазвичай поставляється з операційною системою Windows, так як є розробкою компанії Microsoft. Використовує протокол TCP для стримінгу каналів, а також UDP при роботі з медіаданими.

Рисунок 1.1 – Інтерфейс Skype

До переваг цього застосунку можна віднести:

— більшість потрібного функціоналу безкоштовна;

— наявні автоматичні субтитри;

— легкий у налаштуванні.

Недоліками є:

— необхідність встановлювати на персональний пристрій для отримання усього спектру функціоналу;

— погана оптимізація;

— відсутня робота с перекладом;

— відсутня робота по трансформації голосу.

1.3 Програмне забезпечення Zoom

З моменту, як було оголошено карантин у світі, капіталізація компанії Zoom виросла у декілька разів. На даний момент – це найпопулярніший та найкращий додаток у сфері конференц-рішень. Zoom – пропріетарна програма для організації відеоконференцій, розроблена компанією Zoom Video Communications [2](рис 1.2). Працює на базі протоколу UDP та передає пакети даних у зашифрованому вигляді за допомогою алгоритму криптування AES-256.

Рисунок 1.2 – Інтерфейс Zoom

Переваги:

— підтримка найбільшої кількість одночасно підтримуваних користувачів;

— надійне підключення та якість конференцій;

— зручний і нескладний інтерфейс;

— велика кількість функціоналу, включаючи роботу з аудіоканалом та трансляцію тексту;

Недоліки:

— обмежені можливості та зручність використання у безкоштовному плані;

— використовує велику кількість ресурсів;

— має певні проблеми з продуктивністю додатку;

— відсутність перекладу субтитрів та їх озвучування.

1.4 Google Translate API

Google Translate API – це веб-служба компанії Google, призначена для машинного (автоматичного) перекладу слів, частини тексту або веб-сторінки на іншу мову [3]. Підтримує зчитування та переклад записаного аудіотреку. Інтерфейс сервісу зображено на рисунку 1.3. Один із найбільш технічних додатків для перекладу. Використовує HTTP протокол для запитів на переклад.

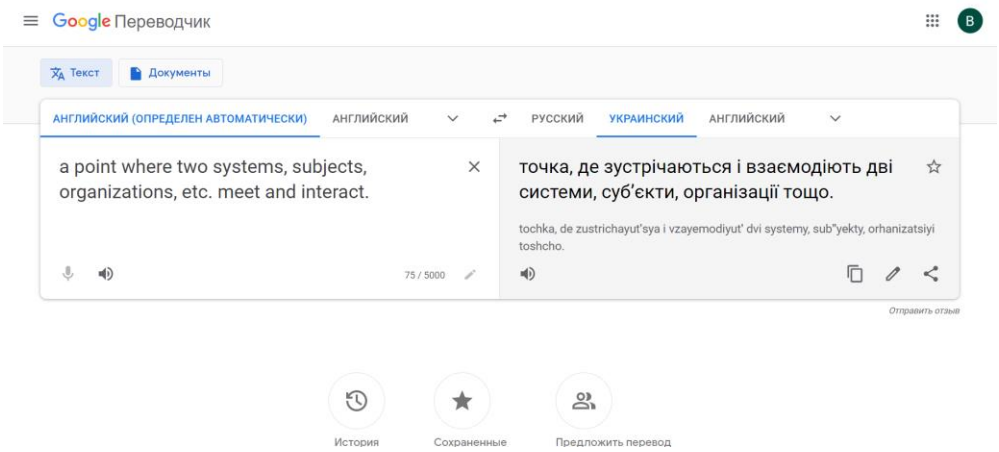


Рисунок 1.3 – Інтерфейс Google Translate

Переваги:

— висока точність перекладу;
— швидкість перекладу;
— можливість записувати аудіофрагменти;
— встановлення додатку не є обов’язковим;
— озвучує переклад.

Недоліки:

— не є засобом для відеокомунікації.

1.5 Гарнітури-перекладачі

Гарнітури-перекладачі – це пристрої, зазвичай у вигляді навушника, котрі розпізнають іноземну мову та перекладають її на іншу. Пристрої для перекладу з однієї мови на іншу існували і раніше, але не користувалися популярністю через велику кількості смислових помилок. З появою нових технологій вченим вдалося вдосконалити процес.

В результаті, на ринок стали виходити сучасні навушники, здатні правильно перекладати мову в режимі реального часу на 40 мов світу. Вони допоможуть подолати мовні перепони і забезпечують комфорт при спілкуванні з іноземцями в подорожах або на заходах.

Такі прилади зазвичай використовують специфікацію Bluetooth для трансляції аудіоканалу на бік персонального пристрою.

До прикладу таких пристроїв можна віднести: TimeKettle [4](рис. 1.4), WT2, Waverly Labs.

Рисунок 1.4 – Гарнітура для перекладу TimeKettle M2

Переваги:

— висока точність перекладу;
— велика кількість підтримуваних мов;
— озвучує переклад.

Недоліки:

— необхідно купувати пристрій;
— необхідне встановлення додатку на телефон;

Отже, під час аналізу було виділено необхідні риси для забезпечення успішного майбутнього додатку для відеоконференцій зі спеціалізацією на роботу з мовою, такі як: робота з медіапотоками даних, необхідність встановлення додатку, зручний протокол для інтеграцій, підтримка багатьох мов, переклад озвученого тексту, озвучування тексту. Для наочності, в таблиці 1.1 наведено всі переваги кожного із рішень що було описано вище.

Таблиця 1.1 – Порівняння аналогів

Назва	Необхідність встановлення додатку	Підтримка багатьох мов	Переклад розпізнаного тексту	Озвучування результатів	Стримінг медіа даних
Skype	-	-	-	-	+
Zoom	+	-	-	-	+
Google Translate	+	+	+	+	-
Гарнітури-перекладачі	-	+	+	+	-

Висновок до розділу

В даному розділі було проаналізовано та описано складові елементи, котрі притаманні більшості продуктів для відеокомунікації на ринку, такі як: відео- та аудіодзвінки, запис дзвінку, презентація екрану, текстовий чат, субтитри та інше. Також у розділі було оглянуто, з якими складнощами можна зіткнутись при розробці систем даного класу. До таких відносять: роботу з якістю контенту, який передається, безпеку підключення та обміну трафіком, роботу зі стандартами браузерів.

Так як у роботі розглядається відеокомунікація зі спеціалізацією на роботу з мовою та перекладом, то було описано та розглянуто такі аспекти як: наявність субтитрів, робота з перекладом, озвучування тексту та інші функціональні вимоги. Також, важливо було розібрати та описати нефункціональні вимоги, до яких належать: необхідність встановлення на персональний пристрій, швидкість затримки до зчитування, швидкість перекладу тексту, підтримка кількості учасників, ціна масштабування, особливості способу трансляції.

На відповідність вище поставленим вимогам було розглянуто декілька систем-аналогів, котрі пов’язані з поставленою проблематикою. До таких систем

відносимо: Skype, Zoom, Google Translate, гарнітури для перекладу. Ці додатки можна поділити на ті, що спеціалізуються саме на відеоконференціях, і ті, що націлені на роботу з мовою та перекладом.

Таким чином, необхідно звернути увагу на особливості існуючих аналогів, та об’єднати їх характеристики для побудови системи для відеоконференції зі спеціалізацією на роботу з людською мовою та перекладом.

2 РОЗРОБКА АРХІТЕКТУРИ

Для того, щоб вірно побудувати архітектуру, потрібно чітко визначитися з функціональними модулями системи. На основі аналізу функціональних та нефункціональних можливостей систем аналогів, сформульовано вимоги до системи що розроблюється. Комплексна система, яка розглядається у роботі, включає в себе багато компонентів, що мають додаткові залежності зі сторони публічних сервісів. За основу візьмемо модель взаємодії сервісів за допомогою мережевої комунікації, так як вона дозволяє представити систему, як набір функціональних сервісів, котрі спілкуються один з одним, та можуть працювати незалежно від помилок інших. Кожний з модулів повинен мати чітко визначений інтерфейс взаємодії, для того, щоб бути самостійною системною одиницею та мати потенціал до існування, не знаючи про інші деталі системи.

До таких функціональних модулів можна віднести:

— модуль по роботі з даними користувача та його заходів;
— модуль прийняття рішень;
— модуль графічного інтерфейсу;
— модуль по комунікації;
— модуль по роботі з аудіопотоками;
— модуль перекладу;
— модуль розпізнавання мови;
— модуль статистики.

Для обґрунтування коректної взаємодії цих модулів потрібно розглянути основні бізнес-сценарії, котрі будуть відбуватись у рамках додатку. Виходячи з опису функціональних вимог, варто розглянути два сценарії: сценарій підключення до конференції між учасниками та сценарій транслювання перекладної фрази одного учасниками іншим.

2.1 Архітектура системи для відеоконференцій

Розглянемо сценарій підключення до конференції. Потрібно урахувати момент, що клієнти транслюють інформацію зі своїх медіаносіїв у веб-браузер або мобільний додаток, з цього випливає, що відсутня мережева інформація про клієнта, така як IP-адреса та порти взаємодії.

Алгоритм підключення користувачів до конференції має наступні кроки:

— користувач через клієнта робить запит на підключення до конференції;

— клієнт робить запит на доступ до сесії;

— клієнт робить запит на підключення до серверу оповіщення;

— клієнт підключається до серверу, або робить запит на відновлення підключення у разі помилки;

— клієнт отримує доступ про авторизацію користувача у сесії;

— сервер сесій посилає запит на нотифікацію про нового учасника в сервер оповіщення;

— сервер оповіщення надсилає сповіщення всім іншим учасникам про параметри нового підключення;

— клієнти учасників, отримавши інформацію, підключаються через STUN/TURN сервери.

Даний алгоритм зображено на діаграмі послідовності рисунку 2.1

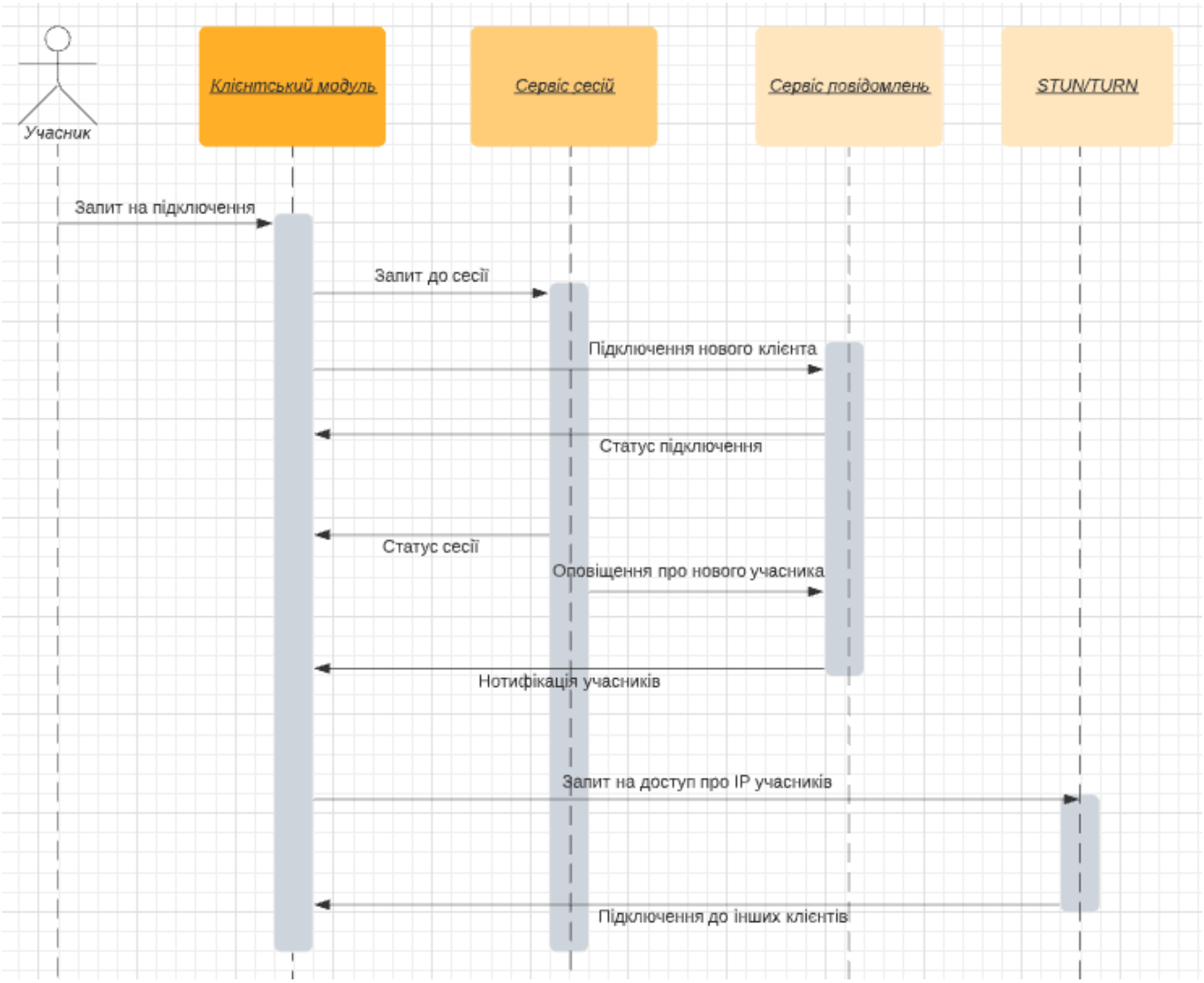


Рисунок 2.1 – Діаграма послідовності підключення учасників

С описаного видно, що нам необхідно мати сервіси, відповідальні за сесії учасників, сервер оповіщення, котрий буде потоково продавати інформацію про події та сервери STUN/TURN для роботи з підключеннями.

В рамках розробляємої архітектури частину системи відповідальну за зв’язок між клієнтами представлено на рисунку 2.2.

Рисунок 2.2 – Архітектурна схема частини відеоконференції

Таким способом можна забезпечити комунікацією між програмними компонентами таким чином, щоб можна було інтегрувати взаємодію з іншими функціональними модулями.

2.2 Архітектура функціональної частини

Наступний критичній бізнес-сценарій для системи є саме робота з перекладом у реальному часі під час конференції з декількома учасниками. Наступний сценарій включає в себе оптимізації по одночасній роботі з перекладом та аспекти трансляції результату до інших учасників:

Отже, алгоритм перекладу аудіопотоку на іноземну мову виглядає наступним чином:

— клієнт робить запит на сервер сесій про початок трансляції аудіопотоку для перекладу;

— сервер відповідає, чи доступна така можливість для поточного клієнта у даний момент часу;

— у разі якщо доступна, то сервер сесій блокує цю можливість для інших, поки поточний користувач повністю не завершить операцію;

— клієнт починає зчитувати медіапотік із каналів зв’язку з додатковою затримкою для забезпечення вірного завершення обробки відрізку аудіотреку;

— після фіксації, аудіопотік направляється в сервіс по роботі з голосом;

— сервіс по роботі з голосом транслює отриманий аудіопотік у сервіс по роботі з розпізнавання голосу в текст;

— паралельно, сервіс по роботі з голосом зчитує тембр та аудіоналаштування отриманого потоку;

— сервіс по розпізнаванню голосу в текст відсилає запит до провайдера розпізнавання. після розпізнавання отриманий текст повертається до серверу роботи з голосом;

— розпізнаний текст відправляється на сервіс по роботі з перекладом. отриманий текстовий переклад повертається до сервісу по роботі з голосом;

— результуючий текст озвучується та до нього засовуються аудіофільтри та додаються метадані;

— результуючий аудіострім проходить назад через сервер сесій та транслюється іншим учасникам через сервер оповіщення;

— відбувається зняття блокування і надається можливість наступним користувачам до перекладу.

Даний алгоритм надає можливість достатньо швидко та точно опрацьовувати медіадані для перекладу. Через повне блокування на переклад відкидається можливість паралельного спілкування декількох людей, що може буде відмінним, порівняно зі спілкуванням у реальному житі. Для часткового вирішення цієї проблеми, пожертвувавши деякою чистотою отриманого результату, можна впровадити наступні кроки покращення алгоритму:

— після того, як аудіопотік було повністю зчитано та відправлено у сервіс по роботі з голосом, знімається блокування для доступу до функції перекладу інших учасників;

— у разі, якщо текст ще не був перекладений, то наступний вхідний аудіопотік переміщається у чергу та оброблюється тоді, коли звільнюється сервіс по роботі з голосом у рамках поточної сесії;

— результуючий текст озвучується по готовності, у асинхронній манері.

У рамках даного сценарію (див. додаток А) видно, що для функціонування потрібно мати сервіси по роботі з розпізнаванням тексту, перекладу, та акумулюючий сервіс по роботі з голосом, котрий буде збирати дані, сам їх аналізувати та видозмінювати за необхідності. Основним сервісом по роботі є саме частина прийняття рішень, так як вона відповідає за блокування доступу до функціоналу для інших користувачів, а також відповідає за те, на яку мову потрібно робити переклад.

Також у системі присутні такі моменти, як запити на отримання інформації про особисті дані клієнта, адміністрування їх, а також інформації про конференції та заходи, проте вони не є складними і вирішуються HTTP запитом напряму до сервісу клієнтських даних.

Кожний сервіс збирає свої метрики, і для того, щоб не робити додаткове навантаження на їх зберігання чи на надсилання до ресурсу, котрий збирає їх, варто створити сервіс, котрий буде займатись тим, що буде сам у проміжку деякого інтервалу робити запит на метрики та зберігати їх. Також, для кращого аналізу та доступу до отриманих даних потрібно додати окремий сервіс саме по роботі з агрегаціями, графіками та іншими інструментами аналізу.

Проаналізувавши вище описані сценарії та функціональні та нефункціональні вимоги, розроблено наступну архітектуру (див. додаток Б).

Як видно з діаграми, систему можна представити у вигляді 4 функціональних компонентів системи, а саме:

— клієнтський модуль, котрий відповідає саме за зв'язок між учасниками;

— модуль прийняття рішень. котрий приймає рішення по доступу до функцій під час сесії, а також налаштування мови перекладу різних учасників.;

— модуль роботи з голосом, а саме робота з розпізнаванням, перекладом та трансформацією;

— модуль статистики, котрий складається з сервісів збору статистики та роботи зі зібраними даними;

У дану архітектуру закладені такі переваги, як:

— незалежне масштабування кожного компонента;

— динамічна конфігурація кожного сервісу;

— асинхронна робота деяких сервісів;

— легкість розширення підтримки нового функціоналу та провайдерів;

— можливість поділу відповідальності за функціональні частини між різними командами розробників.

Серед недоліків можна виділити:

— складність розуміння залежності між серверами оповіщення та сесій;

— розробники повинні мати розуміння концепцій мікросервісів;

— нові функціональні модулі повинні дотримуватись строгих інтерфейсів існуючих;

— відсутність можливості асинхронного розгортання.

2.3 Вибір архітектури програмних компонентів

На сьогоднішній день, існуючі системи для обміну інформацією являють собою складні комплексні системи, котрі включать в себе багато функціональних компонентів та, залежно від своєї програмної архітектури, потребують певні навички зі сторони розробників, а також ресурсів для підтримки працездатності системи в рамках нефункціональних вимог. Не дивлячись на існування сучасних підходів, таких, наприклад, як serverless архітектура та інші, найбільш вживаними є саме монолітна архітектура та архітектура мікросервісів. Кожна з них має свої особливості, котрі потрібно порівняти на доцільність використання в системах конференцій зі спеціалізацією по роботі з мовою.

2.3.1 Монолітний тип архітектури

Архітектурний тип системи «моноліт» є дуже відомим та більшість додатків написана саме за ним, навіть якщо розробники не знають про це. Така система характеризується концентрацією всього функціоналу в рамках однієї програмної

одиниці. Такі додатки гарно підходять для демонстрації прототипу системи та концепції. В свою чергу, потребує меншу кваліфікацію та менше навичок з проєктування зі сторони команди розробки. Різновид інструментів – це те, що дозволило більшості веб-додатків мати монолітну структуру. Монолітні додатки мають велику перевагу у ресурсному та фінансовому підтримуванні, особливо при використанні маленькою командою розробників, мінуси також присутні і вони вагомі.

Монолітна архітектура – це така архітектура, в якій програмний додаток призначений для роботи, як єдиний автономний пристрій. Компоненти в монолітній архітектурі взаємопов'язані та взаємозалежні, що призводить до міцно пов'язаного коду [5]. У більшості великих корпорацій існують основні додатки з даним типом, тому що зв’язаність компонентів та складність бізнес-логіки функціоналу не дозволяють розширювати додатки для масштабування, і витрати на рефакторинг або створення нового продукту з нуля дуже великі, порівняно з підтримкою існуючої системи монолітного типу.

До сучасних проблем, з якими більшість бізнесів зіштовхується при роботі, відносять пошук помилок та масштабування при зростанні активних користувачів. Даний тип архітектури не підходить через свою складність та кількість програмного коду в межах одного функціонального модуля. Збільшення часу пошуку помилок зменшує час розробника на створення нового функціоналу, і в свою чергу створює більші витрати на команду розробки.

Серед переваг виділяють:

1) простота розробки (характеристика архітектури дозволяє отримати перевагу у легкості створення, тестуванні та розгортанні. Якщо продукт має невелику кількість функціоналу, то швидкість розгортання дуже висока. Система масштабується тільки вертикально, що дозволяє не турбуватись про інфраструктурні моменти. Такі системи мають лише одну точку відмови, що дозволяє зменшити час на проходження тестів).

2) проста інфраструктура (маючи все в одному додатку, легким стає повторно використовувати налаштовування програмних модулів, які

застосовуються в інших місцях. Віртуальні машини або сервери у хмарному середовищі, де розгортається система, мають меншу кількість залежностей, що зменшує можливість відмови системи та витрати на розгортання та підтримку таких систем);

3) продуктивність (через комунікацію, котра відбувається на рівні процесу, відбувається оптимізація роботи програмною платформою або операційною системою. Немає необхідності використання систем, котрі вирішують проблеми синхронізації такі, як: розподілені кеші, через те, що потрібну інформацію можна утримувати в рамках пам’яті додатка).

Недоліки, котрі важливо ураховувати при проєктуванні виділяють:

1) відсутність повторного використання (монолітні програми не використовуються повторно, тому що вони використовують інструменти, які надаються спеціально для інструментів та підходів цільової платформи. Оскільки він спирається на всю екосистему, в якій він був розроблений, неможливо повторно використовувати функції та інтегрувати їх до інших проєктів. Це означає, що функція не буде працювати поза програмою, в якій вона була розроблена);

2) проблеми при масштабуванні (проблема починається з масштабування цих систем, оскільки моноліти виконують усі бізнес-операції, зберігають дані та проводять сеанси обробки. Горизонтальне масштабування не може гарантувати цілісність і стійкість до втрати даних, а вертикальне масштабування, у свою чергу, є дуже складним для великих підприємств та їх продуктів, оскільки потрібен час, щоб переналаштувати або придбати нове обладнання, якщо підприємство не використовує послуги постачальника хмарних послуг);

3) чорний ящик (цей термін описує систему, котра невідомо як працює усередині. Зі зростанням кодової бази та підключенням до екосистеми технологій, що використовуються, компоненти починають ставати складними та взаємопов'язаними, тому існуючі модулі та процеси необхідно враховувати при розробці нових незалежних функцій. У таких випадках найбільш поширеним рішенням є розширення чи перевизначення існуючого програмного інтерфейсу. Це

не дозволяє використовувати платформу за прямим призначенням для вирішення бізнес-завдань);

4) відсутність можливості вибору технологій незалежно від платформи (багаточисленні інструменти та платформи пропонують широкий спектр функцій, тому простого вибору початкової платформи для розробки може бути недостатньо для вирішення вашої конкретної майбутньої задачі. Додавання нових технологій спонукає розробників розширювати систему, переписує системні модулі або прийняті архітектурні рішення. Проблемою є використовувати наявні інструменти, які не можуть використовуватися для конкретної цілі, через несумісність з цільовою технологію або дорого підтримуються протягом часу. Вирішення є написати свою власну реалізацію та підтримувати її, що є складним та затратним);

5) система контролю версій (при розробці продукту у великій команді може бути складно реалізувати нові функції, тому що необхідно підтримувати поточну версію кожного розробника. Для клієнтів це відбувається неявно, і нам потрібно перезапустити систему та запустити її на хостингу, щоб нова функція запрацювала. Це дуже важливо для систем, яким потрібно працювати 99% часу).

Загалом, моноліти слід розглядати, коли потрібно протестувати певну гіпотезу, якщо вам не потрібно працювати з паралельними обчисленнями великих масивів даних або над проєктом не будуть працювати великі команди розробників, котрим потрібно підтримувати багато функціоналу в одному місці. Простоту такої архітектури зображають у вигляді схем, як на рисунку 2.3.

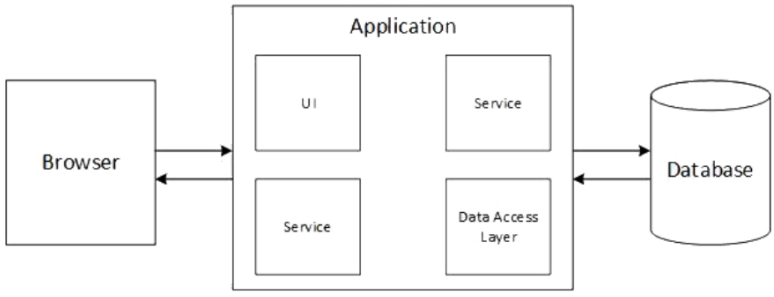


Рисунок 2.3 – Схема монолітної архітектури

Успіх або поразка продукту невідомі, тому ви можете почати з монолітної архітектури, але внутрішні елементи програмного модуля поділяємо на рівні, щоб створити слабкі зв’язки всередині моноліту для додаткової функціональності, яку можна перенести до інших типів архітектури через нього.

Для систем, зі специфікою, котра розглядається у даній роботі даний тип архітектури не підходить через неможливості масштабування навантаження на сервери з обчисленнями, такими як переклад голосу в текст та трансформації голосу. Також, для високої якості роботи системи потрібно підтримувати декілька провайдерів функціоналу паралельно, котрі у своїй більшості мають різну програмну платформу.

2.3.2 Мікросервісний тип архітектури

З кожним днем все більше і більше систем намагаються переробити на даний тип архітектури, а саме з монолітної на мікросервісний. Мікросервіси – це різновид сервісно-орієнтованої архітектури, яка використовується для побудови розподілених програмних систем [6]. Сервіси в архітектурі мікросервісів — це процеси, які спілкуються один з одним через мережу для досягнення мети. Ці служби використовують технічно-агностичні протоколи, щоб допомогти вибрати технологію та структуру, а також можуть вибирати внутрішні служби для вирішення конкретних проблем у певних областях. Цей підхід добре працює з гнучкими методологіями розробки та процесами DevOps [7], створеними після того, як мікросервіси набули всесвітньої популярності серед компаній.

Особливими характеристиками мікросервісів, котрі відрізняють даний тип архітектури є:

— бізнес-орієнтована розробка на предметну область;
— абстрактні розподілені інтерфейси, для масштабування;
— націленість на роботу у хмарних середовищах;
— можливість використовувати різні мови програмування та бази даних;
— розгортання додатків у контейнерах;

— можливість версіювання та розподіленого розгортання;

— DevOps процес з моніторингом сервісів.

Прямими перевагами з вище описаних характеристик мікросервісної архітектури є:

1) краща ізоляція несправностей (великі програми можуть залишатися в критичному стані, не реагуючи на збій одного модуля, що дозволяє системі продовжувати функціонувати, за винятком деяких функцій, які тимчасово не відповідають потребам бізнесу);

2) можливість вибору технологій (мікросервіси забезпечують гнучкість тестування нових технологічних стеків в окремих сервісах за потреби. Проблем із залежностями не так багато, і скасувати зміни набагато простіше. Менше коду надає кожній службі більшу гнучкість. Оскільки можливим є вибрати технологію відповідно до потреб, великі компанії мають багатопрофільні команди розробників, які вирішують конкретні проблеми за допомогою відповідних інструментів);

3) виокремлення бізнес контекстів (розробники мають можливість надавати функціональність окремим мікросервісам, які обробляють роботу з невеликою кількістю кодової бази, що, у свою чергу, легше підтримувати та використовувати);

4) просунуте розгортання (менша база коду та обсяг пропорційні швидшому розгортанню системи, тому ви можете скористатися перевагами безперервного розгортання, мати різні версії одного сервісу та збалансувати трафік для детального тестування нових функцій);

5) гарна масштабованість (оскільки сервіси не пов’язані, ця архітектура дозволяє легко масштабувати найбільш необхідні сервіси вчасно, щоб підтримувати та впоратися з великим навантаженням на систему, на відміну від повної програми. Якщо зробити це правильно, можна заощадити гроші. Отже, великі підприємства мають індивідуальних експертів, щоб автоматизувати розгортання систем або переконатися, що все працює правильно).

Також, наявні недоліки, котрі буде складно покрити погано кваліфікованими спеціалістами, а також недоцільним використанням у рамках невеликих продуктів. До таких недоліків можна віднести:

1) складність розробки (найбільшим недоліком архітектури мікросервісів є її проєктування та обслуговування як цілісного продукту. Висока вартість проєктування та вирішення проблем може бути дуже важливою для невеликих команд. Крім того, готові рішення для вирішення проблем інфраструктури хмарних провайдерів вимагають додаткових витрат та досвіду команди);

2) швидкість розробки (підтримка системної інфраструктури та методів зв’язку може займати найбільше часу у розробника під час випуску нового сервісу, а складність розгортання системи може бути значною, якщо у вас немає експерта, який оптимізував це завдання);

3) безпека системи (коли спілкування відбувається за допомогою повідомлень, цей трафік може бути перехоплений і замінений. Питання безпеки також вимагають додаткових інвестицій в інфраструктуру системи та досвіду команди розробників).

Система, котра базується на основі даної архітектури, може складатись з декількох бізнес-контекстів, у котрому кожний сервіс має свій екземпляр бази даних. Не дивлячись на складну систему зі сторони серверу, найчастіше такі системи мають лише один клієнтській застосунок, котрий запускається у браузері, мобільному пристрої або персональному комп’ютері [6]. На рисунку 2.4 видно, як зазвичай виглядають такі системи.

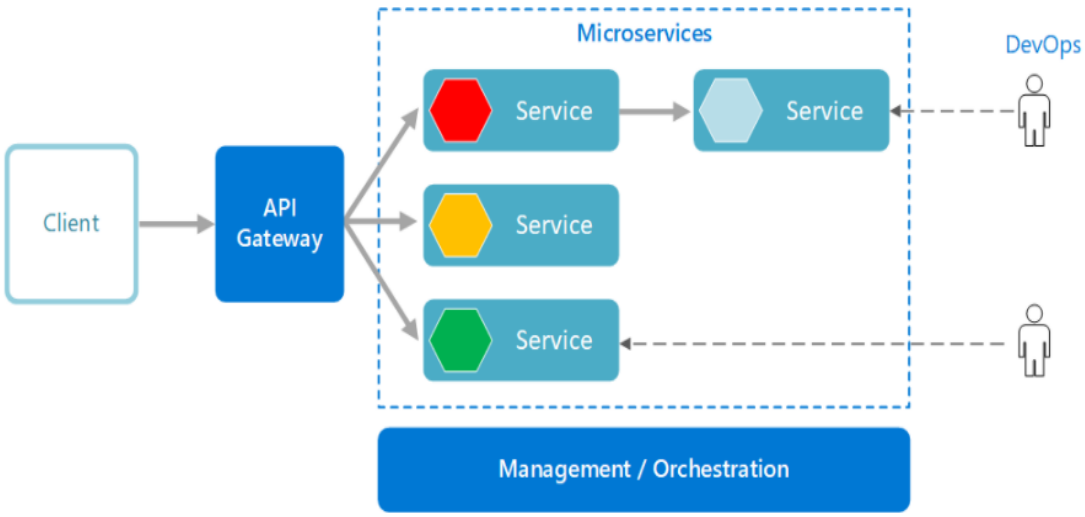


Рисунок 2.4 – Схема мікросервісної архітектури

Розглянувши мікросервісний тип архітектури, та порівнявши його з монолітним, можна зрозуміти, що для системи, котра розглядається у роботі, мікросервіси є найкращим варіантом через наступні моменти:

— необхідність підтримувати декілька провайдерів перекладу, розпізнавання мови та трансформації каналів одночасно;

— підтримувати можливість регулювання ресурсів кожної окремої частини системи;

— програмні реалізації методів роботи можуть бути на таких мовах програмування як Python, JavaScript та .NET;

— необхідність підтримки розгортання в хмарній інфраструктурі.

Висновки до розділу

У даному розділі було запропоновано архітектуру програмного забезпечення комунікації у реальному часі з можливістю роботи з голосом, а саме переклад тексту аудіотреку для подальшого озвучування іншим учасникам конференції.

Було описано критичні для системи бізнес-сценарії та на їх основі побудовано схему архітектури.

У системі можлива поява колізій при одночасному використанні аудіоканалу, що на даний момент вирішується за допомогою модуля прийняття рішення про визначення пріоритету про надання доступу до аудіоканалу.

Запропоновано механізми для компенсації часу затримки функціоналу перекладу для інших учасників або автоматичне визначення пауз.

3 МЕТОДИ ТА ЗАСОБИ ДЛЯ ПОБУДОВИ СИСТЕМИ

3.1 Загальні положення

При взаємодії віддалених користувачів важливою є саме відео- та аудіо комунікація, котру потрібно підтримувати на високому рівні, так як такий спосіб обміну інформацією є найбільш природною формою спілкування. Вагомим фактором виступає саме робота з аудіо. Критерії оцінки якості передачі досить високі, тому що користувачі звикли до якості рухомого зображення, котре ми отримуємо через телевізійні системи, а якість передачі голосу повинна бути на рівні телефонії. Інструмент для конференцій зі спеціалізацією по роботі з перекладом має додаткове навантаження на канали зв'язку, тому також важливо правильно розділити зони відповідальності елементів системи та прорахувати варіанти до масштабування.

Для системи з такою специфікою, котра розглядається у даній роботі, з дотриманням функціональних та нефункціональних вимог, потрібно розглянути методи, котрі пов'язані з такими елементами системи, як: сервіс по роботі з медіапотоками, сервіс для роботи зі зчитуванням тексту з аудіопотоків, сервіс по роботі з перекладом, інструменти та методи озвучування отриманого тексту, фільтри та налаштовувачі для голосу, інфраструктурні моменти та інше.

3.2 Вибір протоколу для роботи з відео- та аудіотрансляцією

Поставлена проблематикою ціль – описати та розробити систему, котра спеціалізується на роботі з трансформацією медіаканалів. Тому важливо підібрати такий протокол, щоб він був сумісний з більшістю існуючих стандартів та бібліотек. Протоколів існує декілька, та базуються вони на протоколах групи IP, та мають багато спільного. Розглянемо протоколи передачі даних: RTP, RTSP, WebRTC.

3.2.1 Real-time transport protocol

RTP є транспортним протоколом, котрий використовується для передачі даних, таких як відео та аудіо, по мережах за допомогою пакетів через Internet. RTP не включає функції маршрутизації, оскільки для цього застосовується служба UDP. Недоліком є те, що UDP базується на TCP, а той у свою чергу орієнтований на гарантію доставлення пакетів, що додає додаткових накладних ресурсів і зменшує швидкість, що є негативним у ситуації високого трафіку та залежності від часу [8].

Для того, щоб передавати мультимедіа трафік без втрати інформації та тимчасових затримок, RTP оброблює ідентифікацію контенту, нумерацію послідовності пакетів і присвоєння порядку пакетів. Для створення сесій на базі протоколу потрібно реалізувати програмний інтерфейс, котрий буде визначати набір адрес одержувачів даних, а саме адреси мереж та портів. Кожному типу трафіку, відео або аудіо, відповідає своя сесія. Це дозволяє вибирати, який вид трафіку цікавий користувачу під час комунікації.

В RTP-заголовку (рис 3.1) зберігається інформація про впорядкування пакетів даних у часі для подальшого відновлення, а також спосіб шифрування контенту, котрий може змінювати свій розмір в залежності від навантаження на систему [8]. Зазвичай такий функціонал не доступний протоколам транспортного рівня. Пакет складається з медіа інформації, що інкапсулюється в RTP-заголовок, потім - в UDP-заголовок і результат в IP-заголовок. Отримане передається через мережу Internet.

Рисунок 3.1 – Структура пакету RTP

3.2.2 Real-time streaming protocol

RTSP – це протокол прикладного рівня, такий як HTTP або FTP. Особливістю цього протоколу є те, що він надає можливість керувати медіапотоком завдяки командам. Протокол, працює використовуючи протоколи транспортного рівня, такі як RTP, TCP, IP та UDP. Логіка по транспорту даних перекладається на протокол нижчого рівня, а RTSP забезпечує взаємодію між клієнтами та форматами даних, які відправляються.

Кожний медіапотік даного протоколу ідентифікується своїм URL, може бути масштабованим, що надає йому схожості з HTTP. Сервіс RTSP підтримується набором інструкцій, якими обмінюються між собою сервер і клієнт. Вони відсилаються у вигляді RTSP-пакетів, що містять установчі параметри для потоку [9]. При стримінгу даних специфікації зберігаються в файлі дескриптору, котрий також можна отримати за окремим URL посиланням.

Не дивлячись на переваги у доволі невеликій затримці та підтримці більшості IP-камер, протокол більше не використовується для стримінгу відео до кінцевих користувачів.

Протокол підтримує різноманітні типи кодування: для відео це H.265. H.264, VP9 та VP8, а для аудіо – AAC, MP3, Opus, Vorbis [9].

Середня швидкість затримки даного протоколу на стримінгу даних складає 2 секунди.

3.2.3 WebRTC

Основна ідея WebRTC полягає у здійсненні зручного прямого зв'язку за допомогою веб-браузера. Таким чином, можна зменшити накладні ресурси на підтримку серверу для стримінгу медіаданих. Основним обов'язком сервера є налаштування обміну конфігураційними даними між компонентами клієнтського програмного забезпечення, як це зображено на рисунку 3.2.

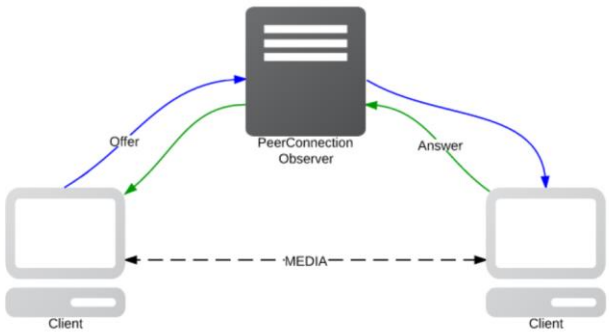


Рисунок 3.2 – Схема використання технології WebRTC

Встановлення зв’язку між двома клієнтами відбувається обміном командами offer та answer [10] «Один – запрошує, а інший – приймає». Клієнти починають обмін даними про користувачів за протоколом ICE. Сервер відповідає лише за обмін ідентифікатором зв’язку та ніяк не впливає на відправку медіаданих, що у свою чергу ізолює несправність лише на стороні клієнта, та дозволяє продовжити комунікацію навіть при проблемах серверної частини продукту.

Це досягається за допомогою інтерфейсу RTCPeerConnection, який дозволяє встановити зв’язок між браузерами [10]. За допомогою цього інтерфейсу можна налаштувати канал зв'язку між клієнтами, що відповідають конкретній системі, передавати дані безпосередньо між браузерами, завантажувати частину серверного програмного забезпечення та дозволяти передавати потоки відео та аудіо між браузерами без подальшої розробки браузера. Формат передачі даних клієнт-сервер за специфікацією може бути обраний розробниками програмного забезпечення.

Для обміну інформації про підключення через протокол ICE необхідно використовувати сервери, котрі відповідають за знаходження мережевої адреси та портів учасника, а саме STUN-сервера. Без цього серверу, можна буде зв’язатись лише у локальній мережі.

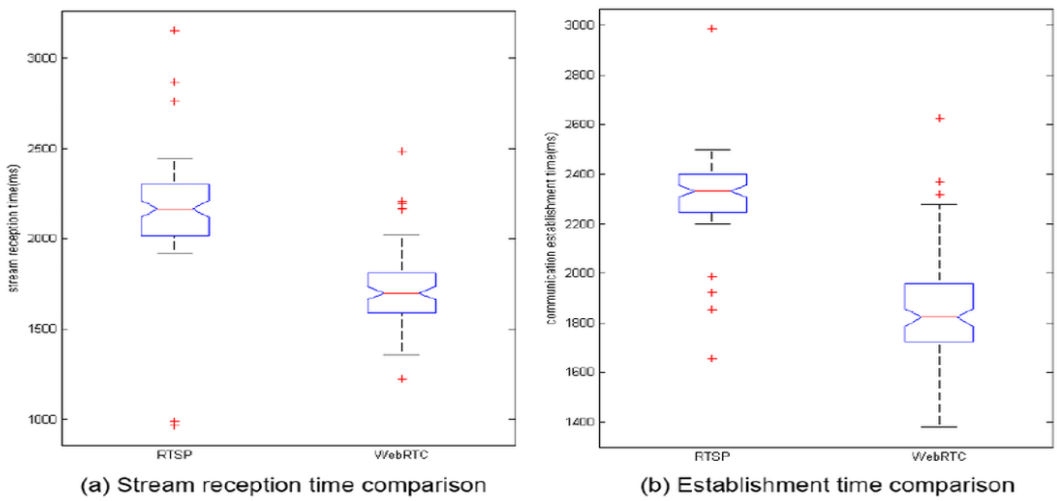
STUN – мережевий протокол, який дозволяє користувачеві визначати свою зовнішню IP-адресу, метод перекладу адреси та порт у зовнішній мережі, пов'язаний з визначенням внутрішнього номера порту [10]. Ця інформація

використовується для створення зв’язку при проходженні трансляції через NAT-маршрутизатори. Процес дозволяє WebRTC отримати публічну адресу, а потім передати її для встановлення прямого зв’язку з іншими системами.

Іноді потрібно використовувати ретранслятора NAT TURN, у ситуаціях коли STUN серверу недостатньо. Він дозволяє хосту або брандмауеру отримати дані з’єднання, між двома точками доступу.

Протокол підтримує типи кодування: для відео це H.264, VP9 та VP8, а для аудіо – Opus, iSAC, iLBC [9]. Протокол є спеціально розроблений під браузери та показує затримку меншу ніж 500 мілісекунд.

Графік середніх показників, отриманих в рамках дослідження [11], затримки підключення користувачів та час для початку стримінгу даних та порівняння їх серед двох протоколів зображено на рисунку 3.3.



(a) Stream reception time comparison (b) Establishment time comparison

Рисунок 3.3 – Порівняння часу затримки для RTSP та WebRTC [11]

Порівнюючи RTSP та WebRTC на критерій затримки стримінгу аудіо- та відеосигналу можна зрозуміти, що WebRTC показує кращі показники, а саме значення менші ніж за секунду, у той час як RTSP має затримку до 3 секунд.

3.2.4 Критерії безпеки комунікації

Проведемо зіставлення параметрів захищеності голосових інтернет-комунікацій за допомогою WebRTC та RTSP у аспекті засобів шифрування. Розглядаючи захищеність того чи іншого способу комунікацій у мережі інтернет, необхідно звернути увагу на вирішення наступних питань:

— методи та алгоритми шифрування;
— спосіб обміну ключами;
— аутентифікація та авторизація користувачів.

Реалізація даних методів в технологіях, що розглядаються, потрібна для забезпечення захисту аналізованих рівнів.

3.2.4.1 Методи та алгоритми шифрування

Технологія WebRTC не регламентує процедуру встановлення сеансу передачі потокових даних і лише визначає безпосередньо сам процес передачі. Тому процедуру встановлення сеансу та передачі будемо розглядати окремо.

При встановленні сеансу зв'язку пропонується використання різних рішень, зокрема Microsoft пропонує в більшості випадків використовувати захищене підключення WS. З одного боку, це дозволяє забезпечувати безпеку підключення, з іншого боку, це збільшує шанси вдалого підключення, так як багато проксі-серверів відхиляють незашифровані підключення WS. Для забезпечення функціонування WSS використовується TLS – безпека транспортного рівня, що являє собою криптографічний протокол, який забезпечує захищену передачу даних між вузлами в мережі, використовує асиметричну криптографію для автентифікації, симетричне шифрування для конфіденційності та коди автентичності [12]. TLS встановлює тунель, який забезпечує низькорівневий TCP зв'язок на кшталт точка-точка через HTTP проксі-сервер, між WS Secure клієнтом і сервером WS. Іншим варіантом процедури захищеного встановлення сеансу передачі потокових даних у рамках

технології WebRTC є використання HTTPS запитів із тривалим часом очікування (long polling). І тут захищеність також забезпечується за допомогою використання протоколу транспортного рівня – TLS. Таким чином, обидва підходи можуть використовуватися для встановлення захищеного сеансу передачі даних. У цьому захист переданих здійснюється з урахуванням протоколу транспортного протоколу TLS. Для безпечної передачі мови та медіаданих у рамках WebRTC передбачається використовувати традиційний Secure Real-time Transport Protocol (SRTP). Алгоритм шифрування AES (симетричний алгоритм блочного шифрування), який використовується у ньому, дозволяє реалізувати функції криптографічного захисту голосових повідомлень. При цьому важливо, що захист пакетів голосової інформації виконується в режимі реального часу, тобто мало впливає на ключові характеристики передачі послідовності і часу обміну даними.

У разі використання RTSP підходи, що розглядаються в контексті технології WebRTC не можуть використовуватися, це пов'язано з тим, що для встановлення сеансу передачі даних використовується один протокол.

У самому протоколі RTSP не встановлено використання будь-якого алгоритму шифрування, лише вказується на необхідність його використання. Однак через деякий час після опублікування даного протоколу з'явився опис його конкретної реалізації, яка використовується в продуктах компанії Adobe. У цій реалізації, використовується шифрування за допомогою шифру AES зі 128-бітними ключами. Пакети встановлення сеансу передачі потокових даних і безпосередньо передані дані шифруються однаковим чином і однаковими ключами.

3.2.4.2 Способи обміну ключами

Для WebRTC обмін ключами на сигнальному рівні відбувається в рамках протоколу TLS. А на рівні передачі потокових даних можуть використовуватися протоколи, які застосовуються для протоколу RTP. Для генерації та розподілу

ключів шифрування медіаінформації між суб'єктами комунікації можуть використовуватись протоколи SDES, ZRTP, DTLS, дамо їх короткий опис. Опис протоколу SDES дано в RFC 4568 [12]. У рамках цього підходу ключ передається в SIP-повідомленні по сигнальному каналу, одержувач використовує його для шифрування трафіку (при цьому обмін сигнальними повідомленнями повинен бути також захищений). Таким чином, можливість використання протоколу SDES обмежується обов'язковістю захисту з'єднання SIP/TLS (описана вище). Іншим обмежуючим фактором є неможливість використання даного протоколу для забезпечення безпеки «з кінця в кінець», наприклад, при використанні BATC, SDES буде розподіляти ключі між суб'єктом комунікації 1 і BATC і між суб'єктом комунікації 2 і BATC, але не безпосередньо між сторонами, котрі приймають участь у комунікації. Протокол ZRTP розроблений спеціально для VoIP і стандартизований в RFC 6189. Протокол забезпечує безпечну автентифікацію та обмін даними. Під час ініціалізації ZRTP-дзвінка використовується метод обміну ключами Діффі-Хеллмана (криптографічний протокол, що дозволяє двом і більше сторонам отримати загальний секретний ключ, використовуючи незахищений від прослуховування канал зв'язку) [13], який не забезпечує захист від атак типу «людина посередині». Для подолання цієї проблеми з метою аутентифікації застосовується SAS (Short Authentication String) «короткий рядок аутентифікації», що є скороченим уявленням криптографічного хешу отриманих ключів Діффі-Хеллмана. Протокол DTLS для SRTP описаний у специфікації RFC 5764. Він описує обмін голосовим трафіком у режимі «крапка-крапка» з жорсткою фіксацією портів UDP у суб'єктів комунікації. Повідомлення протоколу передаються разом із RTP-пакетами. Для організації сесії учасники комунікації обмінюються повідомленнями. Так як в основі протоколу DTLS лежить TLS (Transport Layer Security), що використовує інфраструктуру відкритих ключів (PKI), то застосування DTLS можливе також тільки за наявності вузлів інфраструктури видачі, зберігання та верифікації сертифікатів суб'єктів комунікації. Варто зазначити, що у відповідності з проєктом стандарту тільки DTLS є обов'язковим для підтримки в інформаційних системах, що

реалізують WebRTC.

У разі RTSP обмін ключами здійснюється у два етапи:

— на першому етапі встановлюється сесія і використовується загальний ключ (симетричне шифрування), який заданий у кодуванні UTF-8, при цьому для перевірки цілісності пакета використовується криптографічна 16-бітна контрольна сума, в рамках встановлення сесії відбувається обмін асиметричними ключами для наступної роботи;

— далі на основі алгоритму Діффі-Хеллмана відбувається встановлення загального секретного ключа, яке стає можливим за рахунок встановленого з'єднання, захищеного від модифікації (але доступного для прослуховування);

— після встановлення загального ключа всі нові дані кодуються за його допомогою.

3.2.4.3 Аутентифікація та авторизація користувачів

Технологія WebRTC ніяк не визначає спосіб аутентифікації та авторизації користувачів, тому дане питання вимагає самостійного вирішення.

У реалізації протоколу RTSP відсутня будь-яка підтримка автентифікації та авторизації. Відповідно до будь-якого правильно закодованого сертифіката відповідь буде вважатися достовірною. У зв'язку з цим при використанні протоколу RTSP потрібно розробити окремий алгоритм з аутентифікації та подальшої авторизації користувача, який буде працювати поверх протоколу RTSP . Таким чином, у разі використання протоколу RTSP питання, пов'язані з обміном ключами та шифруванням даних, що передаються, мають вирішення, однак у не вирішених питаннях, пов'язаних з авторизацією та автентифікацією користувачів, вирішення даного питання в рамках даного протоколу має бути виконане на наступному етапі.

Таким чином, зі зіставлення параметрів безпеки описаних технологій видно, що для безпосереднього шифрування використовується алгоритм AES у тій та іншій технології, проте відрізняється механізм обміну ключами. Так для

технології WebRTC на даний момент регламентується використання протоколу DTLS (додатково можуть використовуватися інші алгоритми) для обміну ключами, в той час як для RTSP передбачається власний алгоритм. У той самий час для WebRTC відсутня регламентація протоколу встановлення сеансу зв'язку, тому необхідна розробка самостійного рішення щодо встановлення сеансу. У той же час та інша технологія не дозволяє організувати процедуру автентифікації та авторизації користувачів, що беруть участь у комунікації, і цей процес має бути розроблений самостійно.

Дослідження показало, що в рамках даних технологій для захисту від цієї загрози використовується шифрування алгоритму AES. Однак не регламентується протокол встановлення сеансу, який повинен розроблятися окремо і використовувати самостійні механізми захисту.

За вище описаними характеристиками протоколів, стало зрозуміло, що найкращим вибором для майбутньої системи буде саме WebRTC.

3.3 Інструменти для перетворення голосу в текст

Реалізація системи розпізнавання мови – процес дуже складний, працемісткий та ресурсномісткий. Тому, такі системи постачаються у двох різних виглядах: або у вигляді бібліотеки, котру можна інтегрувати у свій додаток, або у вигляді хмарного сервісу з публічним API. Зазвичай, бібліотеки для озвучування мають гіршу точність зчитування тексту, проте вони є значно дешевші у використані. Публічні API розробляються компаніями на основі штучного інтелекту для підвищення тексту отриманого с аудіотреку, проте використання таких сервісів потребує багато ресурсів, у тому числі фінансових. Для вирішення цих проблем можна інтегрувати будь-який сервіс, який буде задовольняти нефункціональні вимоги такі, як: точність розпізнавання, швидкість та формат, у якому можна транслювати аудіотрек. Порівнявши існуючі сервіси за вище описаними критеріями, можна виділити з гарно описаною програмною документацією: Google Cloud Speech-to-Text, AssemblyAI, Amazon Transcribe. Ці

продукти підтримують більшість сучасних форматів аудіо та велику бібліотеку іноземних мов. При виборі варто звернути увагу на ціну використання. Деякі пропонують оплату за щохвилинне використання, а інші – підписки на сервіси. Моделі розпізнавання кожного з них схожі, тому точність залежить лише від довжини аудіотреку та часу розпізнавання.

3.3.1 Web Speech API

Web Speech API – це бібліотека, котра вбудована Google Chrome, та дозволяє працювати с голосом. Web Speech API надає дві різні сфери функціональності - розпізнавання мовлення та синтез мовлення, які відкривають нові цікаві можливості для доступності та механізмів керування [14].

Розпізнавання слів включає в себе отримання слів через пристрій мікрофона, яке потім перевіряється службою розпізнавання слів за списком граматики (в основному, словника, який ви хочете розпізнати в конкретному додатку). Коли слово або фраза успішно розпізнані, вони повертаються, як результат (або список результатів) у вигляді текстової строки, і в результаті можуть бути ініційовані подальші дії.

Технологія має великий потенціал, проте наразі наявна тільки у браузері Google Chrome.

3.3.2 Google Cloud Speech-to-Text

Google Cloud Speech-to-Text – один з прикладів хмарних сервісів, котрі мають просунуту модель машинного навчання, що дозволяє вигравати у якості та швидкості своїх конкурентів, як наприклад сервіс Amazon Transcribe. Перевагою деяких конкурентів є те, що вони пропонують деяку кількість хвилин запису протягом місяця, в свою чергу – Google Cloud надає лише пробний період, а потім приблизно 10 доларів США за кожні 500 хвилин використання. Дослідження 2020 року показують, що сервіс від Google має точність 84 відсотки [15].

Сервіс від гугл має детальну документацію публічного API, котре працює у двох варіантах: система може завантажити файл аудіозапису на сторону певного сховища, та передати посилання на доступ до нього за допомогою HTTP виклику, або напряму транслювати аудіопотік до сервісів розпізнавання.

3.4 Інструменти для перекладу тексту

Інструменти для перекладу тексту базуються на своєму вбудованому словнику та моделі штучного інтелекту. Тому, як у випадку з роботою з аналізом аудіопотоку та сказаного тексту, важко буде знайти бібліотеку, котра без додаткових тренувань моделі та інших зусиль дасть гарний результат.

Абсолютним лідером у продуктах даного типу займає Google Translate. У 2017 році було проведено опитування, у котрому розглядалась точність роботи цього сервісу, у результаті дослідження було отримано цифру точності 85% [16].Серед конкурентів він має найбільшу швидкість перекладу, перевищуючи їх у декілька разів. Має підтримку більше ніж 100 мов. Дозволяє безкоштовно відправляти до 50 запитів на день. Існують реалізовані бібліотеки по роботі з ним на сучасних мовах програмування, таких як: C#, Java, JavaScript, Python. Також, можна відсилати запити на сервер за допомогою HTTP.

3.5 Інструменти для озвучування тексту

У даному розділі є багато інструментів для вирішення озвучування тексту. Існують як і публічні хмарні сервіси, так і бібліотеки. Такі технології поділяються на два типи: ті що, працюють на стороні клієнта, а тобто браузера, та ті що, на стороні сервера, що дозволяє транслювати аудіопотік до клієнтів.

Прикладом бібліотеки для озвучування тексту в браузері є Speech Syntesis API[14]. Цей функціонал є частиною Web Speech API, проте він є доступний у більшості сучасних браузерах. Бібліотека дозволяє вибирати налаштування для

тембру голосу, швидкість та інші налаштування. Підтримує багато мов та різних акцентів.

Зі сторони серверу, варто розглядати інтеграцію з публічними сервісами, так як озвучування тексту також базується на моделях машинного навчання. Бібліотеки приймають в себе текстовий запит або документ, та перетворюють його в аудіопотік. Мають гнучке налаштування, як по швидкості, так і по тембру голосу. Підтримують багато мов, та можуть бути встановлені на приватний сервер. Отриманий аудіопотік можна відсилати на сторону клієнта за допомогою відкритого каналу або WS протоколу. До таких сервісів можна віднести Amazon Polly, NaturalReader та Text-to-Speech сервіси від компаній Google та Microsoft.

3.6 Робота зі сховищами даних

Зазвичай, системи, котрі працюють з користувачами повинні зберігати інформацію про них, бізнес-процеси та стан елементів системи. У системах зі специфікою по роботі з конференціями основними бізнес-моделями є інформація про заплановані зустрічі, посилання на конференції з метаданими, інформація про користувачів, налаштування та інше. Дані під час роботи не є релевантними довгий період часу та залишаються актуальними лише під час сесії. Проте, такі дані можуть зберігати у довгостроковій перспективі для аналізу та обробки, задля отримання інформації про потреби цільових користувачів. Кажучи про специфіку роботи з мовою, багато даних буде отримано з записів тексту конференції. Таку інформацію можна використовувати для навчання систем штучного інтелекту для підвищення точності та швидкості роботи системи.

Описавши вище сценарії отримання даних потрібно, щоб сховище відповідало наступним вимогам:

— підтримання гнучкого доступу до структурованих даних;
— низька затримка при масштабуванні;
— висока відмовостійкість;
— гарна робота з коротко існуючими записами;

— можливість заповнювати дані з різною структурою;

— забезпечити безпеку зберігання даних;

— швидкий доступ до даних на зчитування.

З вище описаних вимог можна швидко зрозуміти, що не існує такої бази даних, котра зможе покрити всі вище описані пункти. В результаті проєктування чимось доведеться жертвувати. Про схожу проблему у нереляційних базах даних навіть було виведено теорему CAP. Назва складається с заголовних букв англійських слів Consistency, Availability, Partition tolerance, що у свою чергу відповідно перекладається як цілісність, доступність та стійкість до відмов. Ця теорема була створена науковцем Ериком Брювером та описує, що будь-яке розподілене сховище даних може надати лише дві з трьох гарантій:

1) Consistency – кожний читач бази отримує найновіші дані або помилку.

2) Availability – кожний запис отримує не помилкову відповідь, без гарантії того, що дані будуть найактуальнішими.

3) Partition tolerance – система продовжує працювати, незважаючи на довільну кількість повідомлень, які відкидаються або затримуються мережею між вузлами.

На рисунку 3.4 видно трикутник вибору з списком існуючих сховищ, котрі забезпечують потрібний вибір.

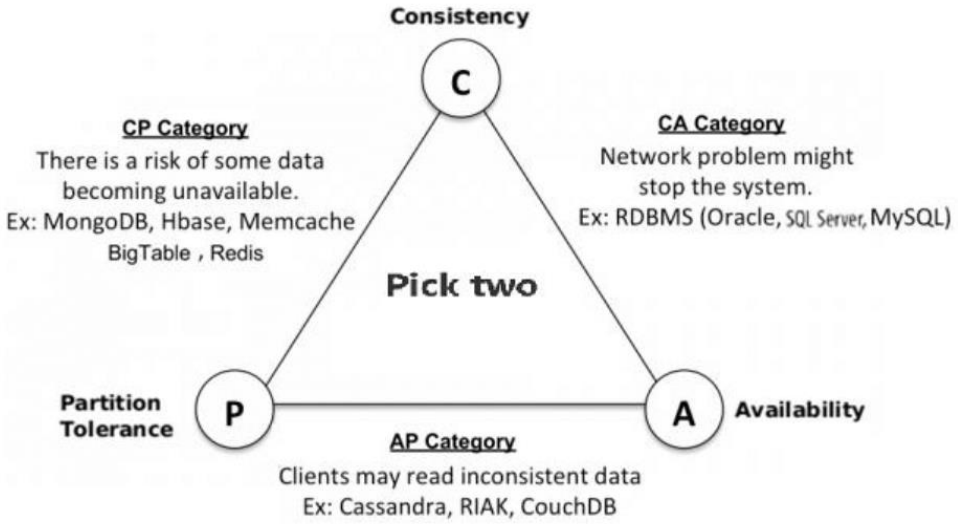


Рисунок 3.4 – Трикутник вибору CAP теореми з прикладами баз даних [15]

Зазвичай ця теорема відноситься саме до розподілених систем, а також баз даних, котрі найчастіше мають можливість зберігати інформацію у довільному форматі. Такі бази даних отримали назву NoSQL, як протиставлення існуючим реляційним базам даних, котрі підтримують мову запитів SQL та оперують строго структурованими даними [18]. Зазвичай SQL бази даних підтримують ACID принципи по роботі з даними.

Потрібно розглянути різницю між NoSQL та SQL сховищами для отримання чіткої картини, яке рішення краще вибрати для поставлених проблем. В рамках таблиці 3.1 видно різницю обраних технологій.

Таблиця 3.1 – Порівняльна таблиця SQL та NoSQL баз даних

Критерії	SQL бази даних	NoSQL бази даних
Модель зберігання даних	Таблиці з фіксованими рядками та стовбцями	Документ: документи JSON, ключ-значення: пари ключ-значення, широкий стовпець: таблиці з рядками та динамічними стовпцями, графік: вузли та ребра
Основне призначення	Загального призначення	Документ: загального призначення, ключ-значення: великі обсяги даних із простими запитами пошуку, широкий стовпець: великі обсяги даних із передбачуваними шаблонами запитів, графік.

Продовження таблиці 3.1

Основне призначення	Загального призначення	Документ: загального призначення, ключ-значення: великі обсяги даних із простими запитами пошуку, широкий стовпець: великі обсяги даних із передбачуваними шаблонами запитів, графік: аналіз і перехід зв’язків між підключеними даними
Схема даних	Строга, заздалегідь визначена	Гнучка
Масштабування	Вертикальне (збільшення масштабу за допомогою більшого сервера)	Горизонтальне (масштабування між серверами-товарами)
ACID транзакції	Підтримуються	Більшість із них не підтримують транзакції ACID з кількома записами. Однак деякі, наприклад MongoDB, роблять
Операції об’єднання	Зазвичай реалізовані та необхідні	Зазвичай відсутні

З таблиці видно, що деякі аспекти кожної із типів бази даних нас можуть зацікавити при реалізації продукту. Завдяки вибраній архітектурі мікросервісів стає можливим використання декількох баз даних в різних функціональних

контекстах та сервісах, для забезпечення актуальних та доцільних нефункціональних вимог системи.

Розглянувши сервіс по роботі з інформацією про клієнтів, котрий буде відповідати за роботу з бізнес-об’єктами по типу інформації про заплановані зустрічі, конференції та мітапи, компанії користувачів та рекламні пропозиції, стає зрозумілим, що ця інформація завжди буде мати стійку структуру та рідко буде змінюватись. Доступ до таких даних на зміну буде з помірним навантаженням. Найчастіше буде акцент на вибірки та запити, котрі можна оптимізувати. Для такої системи адміністрування процесів доцільно буде використати саме SQL базу даних, по типу SQL Server, Oracle чи PostgreSQL. Використавши порівняльні таблиці та відкриту інформацію про особливості цих реляційних баз даних, стало очевидно, що повністю нефункціональні потреби покриває PostgreSQL за найменшу ціну.

Інший важливий модуль, котрий повинен працювати з даними, це сервіс сесій. Цей сервіс тісно пов’язаний з механізмом оповіщення, котрий є важливою частиною роботи з WebRTC, так як цей протокол не відповідає за авторизацію та аутентифікацію клієнтів системи. Цей сервіс буде зберігати та оброблювати дані, котрі не мають довгострокову цінність, а саме інформація про підключення до конференції, медіаматеріали під час конференції, повідомлення у чатах, реакції та інше. Не дивлячись на те, що об’єкти, пов’язані з сесією, втратять свою актуальність через невеликий проміжок часу, доступ до них повинен бути максимально швидкий, так як від цього залежить швидкість роботи системи під час використання користувачем. Для таких цілей доцільно використовувати сховища даних, котрі оперують з даними в операційній пам’яті (RAM) для швидкості роботи. Такі технології дозволяють отримати найшвидший доступ до даних, а також мають можливість зберігати дані в окремий ресурс, якщо в цьому є необхідність. Найкращими представниками таких сховищ є нереляційні бази даних формату ключ-значення Redis та Memcached. Redis є більш доцільним у використані, так як має можливість зберігати інформацію на диск, що дозволяє не втратити інформацію про сесію у випадку відмови сховища.

Решта частина системи буде працювати з NoSQL базою даних MongoDB, так як потрібно зберігати інформацію про події під час конференції, збирати статистику, інформація про повідомлення не в рамках конференції, нотифікації та інше. Така база даних повинна витримувати високе навантаження на запис, що чудово працює у MongoDB при її гарних характеристиках до масштабування. Ця база даних оперує документами, котрі представлені у форматі BSON, що схожа до JSON та має найбільшу підтримку зі сторони товариства розробників серед усіх документо-орієнтованих баз даних.

3.7 Хмарна інфраструктура платформи

Інфраструктура хмарних обчислень – це сукупність апаратних та програмних елементів, необхідних для забезпечення хмарних обчислень. Вона включає в себе обчислювальну потужність, мережу та сховище, а також інтерфейс для доступу користувачів до своїх віртуалізованих ресурсів. Віртуальні ресурси відображають фізичну інфраструктуру з такими компонентами, як сервери, мережні комутатори, пам’ять і кластери зберігання. Хмарна інфраструктура пропонує ті ж можливості, що й фізична інфраструктура, але може надати додаткові переваги, як-от нижча вартість володіння, більша гнучкість та масштабованість.

Інфраструктура хмарних обчислень доступна для приватної хмари, загальнодоступної хмари та гібридних хмарних систем. Також можна орендувати компоненти хмарної інфраструктури у постачальника хмари через хмарну інфраструктуру, як послугу (IAAS). Системи хмарної інфраструктури дозволяють інтегрувати апаратне та програмне забезпечення та можуть забезпечити єдину платформу керування кількома хмарами. Зазвичай, такі платформи складаються з багатьох інфраструктурних шарів та залежать від обладнання, мережі та програмного забезпечення, як це зображено на рисунку 3.5.

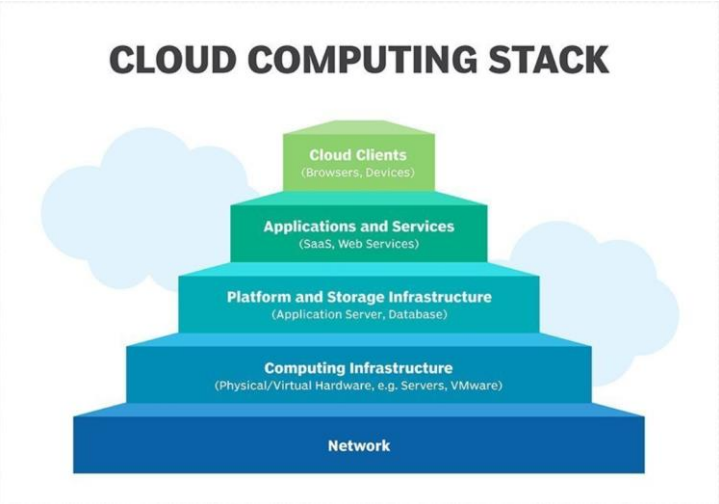


Рисунок 3.5 – Ієрархія складових хмарної інфраструктури

Найпопулярнішими хмарними провайдерами інфраструктури у нас час є Amazon Web Service, Microsoft Azure та Google Cloud Platform. Порівнявши їх характеристики, стає зрозумілим, що провайдер від Google трохи відстає від своїх конкурентів у плані послуг та цін на сервіси. Тому цікаво буде розглянути саме різницю між AWS та Azure Microsoft, котра відрізняється в використанні цільової платформи [19]. Вони мають багато спільного, проте різноманітне зображено в таблиці 3.2.

Таблиця 3.2 – Порівняння AWS та Microsoft Azure

Критерій	Amazon Web Services	Microsoft Azure
Обчислення	У нас є комп’ютери для обчислення, обробки та обчислення даних, і ми можемо масштабуватися до тисяч вузлів обробки за допомогою постачальників хмарних послуг відповідно до наших вимог. AWS	Для обчислювальних цілей Azure використовує віртуальні машини, а для масштабування в значній мірі використовує набори масштабування віртуальних машин, а для керування програмним з

Продовження таблиці 3.2

	використовує Elastic Compute Cloud (EC2), як основне рішення для масштабованих обчислень і для керування контейнером програмного забезпечення за допомогою Docker або Kubernetes, він використовує ECS (сервіс EC2 Container) і використовує реєстр контейнерів EC2.	забезпеченням, у контейнері Docker він використовує службу контейнерів (AKS) і використовує реєстр контейнерів для реєстру контейнерів Docker.
Сховища даних	Сховище знаходиться поруч із основним сервісом для постачальника хмари. AWS використовує S3 (Simple Storage Service), який працює найдовше, ніж Azure, і надає багато документів та навчальних посібників. Він пропонує зберігання архівів на льодовику, архів даних і рідкий доступ S3 (IA).	Azure використовує блоб-об’єкти для зберігання, яке складається з блоків, і ефективно завантажує великі об’єкти. Він використовує Storage Cool і архів зберігання для архівування даних.

Продовження таблиці 3.2

Мережа	Хмарні провайдери пропонують різних партнерів і мережі, які будуть взаємодіяти з центрами обробки даних за допомогою різних продуктів. AWS використовує віртуальну приватну хмару для роботи в мережі та використовує шлюз API для міждомового підключення. AWS використовує еластичний підхід для балансування навантаження під час роботи в мережі.	Azure використовує віртуальну мережу для роботи в мережі або доставки вмісту, а також використовує шлюз VPN для міжмісного підключення. Для балансування навантаження під час доставки вмісту він керує балансувальником навантаження та шлюзом додатків.
Розгортання додатків	AWS також пропонує подібні рішення з Elastic Beanstalk, Batch, Lambda, контейнерним сервісом тощо. Але він не має багато функцій на стороні хостингу додатків.	Однією з переваг хмарних провайдерів є простий процес розгортання програми. Як розробники, ми хочемо розгорнути наш додаток на кількох серверах віртуально за допомогою функцій

Продовження таблиці 3.2

		PaaS. Azure має кілька інструментів для розгортання додатків, таких як хмарні служби, контейнерні служби, функції, пакети, служби програм тощо.
Бази даних	Майже всі постачальники хмарних послуг надають можливість реалізувати базу даних, як у рішеннях SQL, так і в NoSQL. AWS використовує реляційну базу даних, як службу за допомогою RDS, для NoSQL — Dynamo DB, а для кешування — Elastic Cache.	Azure використовує базу даних SQL, MySQL і PostgreSQL для реляційної бази даних, Cosmos DB для рішень NoSQL і Redis Cache для кешування.

Тобто можна зробити висновки, що якщо ми шукаємо IAAS або широкий спектр послуг та інструментів, слід вибрати AWS. Якщо шукаємо інтеграцію Windows або хорошу платформу, як послуги (PaaS) хмарного постачальника, то краще Azure. Саме через ці характеристики та привабливі фінансові показники у роботі будемо розглядати інтеграцію з Amazon Web Services.

Висновок до розділу

У даному розділі було розглянуто технології та методи для побудови системи для відеокомунікації зі спеціалізацією на роботу з мовою та перекладом.

Було детально розібрано особливості кожного проколу передачі даних, а також зроблено акцент на питання безпеки підключення та обміну інформації.

Для основи обміну відео- та аудіопотоками між клієнтами було вибрано протокол WebRTC, через зручність у використані, котрий знімає необхідність у розробці серверного маршрутизатора пакетів даних.

Для роботи з мовою та перекладом, перевага віддається сервісам Google Cloud, таким як Speech-to-Text для роботи з трансформацією аудіопотоку у текст, та Translate API для роботи з перекладом.

Для роботи по озвучуванню тексту та налаштуванню результату аудіопотоку було вибрано підхід для обробки на стороні сервера шляхом транслювання вихідного аудіопотоку на сторону хмарного провайдера Google Cloud Text-to-Speach.

Через складність системи, що розглядається, було розглянуто різноманітні види баз даних, котрі спеціалізуються на вирішені поставлених нефункціональних вимог. До таких технологій було обрано реляційну базу даних PostgreSQL, розподілене сховище даних формату ключ-значення Redis, та NoSQL сховище MongoDB.

Підтримка систем такої складності вимагає багато ресурсів зі сторони розробників, тому було оглянуто варіанти мінімізації затрачених зусиль при виборі інфраструктури розгортання, а саме провайдерів хмарних сервіс, таких як Amazon Web Services або Microsoft Azure.

4 РЕАЛІЗАЦІЯ ПРОТОТИПУ СИСТЕМИ

4.1 Розробка клієнтської частини

Для забезпечення незалежності від цільової платформи та встановлення додатку локально на персональні пристрої було вирішено розробляти клієнтську частину в веб-браузері. Головними програмним об’єктами клієнтської частини є програмні інтерфейси RTCPeerConnection та Web API. RTCPeerConnection – це API, який використовується протоколом WebRTC для створення зв’язку між учасниками конференції для трансляції аудіо- та відеоданих. Web API у свою чергу – це набір програмних інтерфейсів, котрі вбудовані у браузер та дозволяють широко маніпулювати можливостями браузеру.

Об’єкти інтерфейсу RTCPeerConnection являють собою основу канального підключення до інших учасників та підтримують зв'язок протягом усього сеансу, тому потрібно продумати клієнтську архітектуру так, щоб не виникало потреби до перезавантаження сторінки.

Сама технологія дозволяє реалізувати зв'язок, використовуючи лише можливості мови програмування JavaScript та мову розмітки HTML, що у свою чергу може бути покладено на будь-яку архітектуру клієнтських додатків. Проте, якщо розглядати аспект перезавантаження сторінок та необхідність у отримані динамічного контенту у реальному часі, потрібні додаткові ресурси на вирішення цих питань. Саме для таких проблем, було створено підхід SPA (Single Page Application), ідея котрого полягає у тому, щоб використовувати єдину сторінку HTML, а за допомогою засобів JS змінювати DOM-моделі сторінки. Таким чином, під час роботи будуть змінюватись лише ті об’єкти та структура сторінки, що повинні, а такі довготривалі об’єкти у пам’яті, як сам connection, будуть залишатись єдиними та незмінними.

Найрозповсюдженішими інструментами для побудови клієнтських додатків з архітектурним стилем SPA, є фреймворк Angular, котрий підтримується компанією Google та використовує усі можливості мови програмування TypeScript та бібліотека ReactJS, котра складає більшу частину сучасних клієнтських додатків

та підтримується компанією Facebook. Принципіальної різниці у тому, яку технологію вибрати немає, все залежить лише від рівня підготовки та наявності знань у команді розробників. Так як нам важливо, щоб об’єкти підключень залишались постійно у пам’яті, для використання ReactJS, може знадобитись більш детальний дизайн, через особливості реактивної моделі бібліотеки.

4.1.1 Імплементація з’єднання між клієнтами

MediaStream API представляє частину Web API, котра відповідає за медіапотоки. Отримані потоки є синхронізованими. У разі входу камери та мікрофона, синхронізуються аудіо та відеодоріжки. Для отримання використовується виклик методу getUserMedia на глобальному об’єкті браузеру navigator [20]. Метод приймає параметри, на основі котрих вирішує, який потік потрібно отримати та з якими обмеженнями. Отриманий потік можна буде передати в RTCPeerConnection. Для доступу до зовнішньої перефірії браузер запросить доступ у користувача одноразово. Код на мові програмування JavaScript, котрий описує механізм представлений на рисунку 4.1.

```
const openMediaDevices = async (constraints) => {
    return await navigator.mediaDevices.getUserMedia(constraints);
}

try {
    const stream = openMediaDevices({'video':true,'audio':true});
    console.log('Got MediaStream:', stream);
} catch(error) {
    console.error('Error accessing media devices.', error);
}
```

Рисунок 4.1 – Лістинг коду для отримання медіа потоку

Якщо розглядати саме механізм підключення, то кожне однорангове з’єднання обробляється об’єктом RTCPeerConnection. Конструктор для цього класу приймає в якості параметра один об'єкт RTCConfiguration. Цей об’єкт визначає, як налаштовано одноразове з’єднання, і повинен містити інформацію про сервери

ICE, які потрібно використовувати. Як було описано у специфіці роботи WebRTC, обидва клієнти повинні обмінятись offer та answer для успішного підключення. Ми налаштовуємо підписку нашого каналу сигналізації, щоб отримати відповідь на наш запропонований опис сеансу від приймаючої сторони. Такий механізм кодом описаний на рисунку 4.2.

```
const configuration = {'iceServers': [{'urls': 'stun:stun.l.google.com:19302'}]}
// Запит на підключення
const peerConnection = new RTCPeerConnection(configuration);
signalingChannel.addEventListener('message', async message => {
    if (message.answer) {
        const remoteDesc = new RTCSessionDescription(message.answer);
        await peerConnection.setRemoteDescription(remoteDesc);
    }
});
const offer = await peerConnection.createOffer();
await peerConnection.setLocalDescription(offer);
signalingChannel.send({'offer': offer});

// Відповідь на підключення
const peerConnection = new RTCPeerConnection(configuration);
signalingChannel.addEventListener('message', async message => {
    if (message.offer) {
        peerConnection.setRemoteDescription(new RTCSessionDescription(message.offer));
        const answer = await peerConnection.createAnswer();
        await peerConnection.setLocalDescription(answer);
        signalingChannel.send({'answer': answer});
    }
});
```

Рисунок 4.2 – Лістинг коду для підключення двох учасників

Після підключення RTCPeerConnection до віддаленого однорангового вузла можна передавати аудіо та відео між ними. Це точка, де ми підключаємо потік, який ми отримуємо від MediaStream API до RTCPeerConnection. Отриманий медіапотік складається щонайменше з однієї медіадоріжки, і вони окремо передаються до RTCPeerConnection, коли відбувається передача медіаданих до віддаленого користувача. Як це можна реалізувати зображено на рисунку 4.3.

```
const localStream = await getUserMedia({vide: true, audio: true});
const peerConnection = new RTCPeerConnection(iceConfig);
localStream.getTracks().forEach(track => {
    peerConnection.addTrack(track, localStream);
});

const remoteStream = MediaStream();
const remoteVideo = document.querySelector('#remoteVideo');
remoteVideo.srcObject = remoteStream;

peerConnection.addEventListener('track', async (event) => {
    remoteStream.addTrack(event.track, remoteStream);
});
```

Рисунок 4.3 – Лістинг коду для отримання медіа-даних

Для того щоб не імплементувати кожен раз цей механізм на офіційному сайті доступний для завантаження файл adapter [21], котрий представляє собою обгортку над прикладним API.

Для роботи механізму оповіщення, про нових користувачів та події, котрі викликані іншими учасниками під час конференції, також потрібно створити програмний інтерфейс по роботі з нотифікаціями через технологію WS , яка дозволяє відкривати інтерактивну сесію загального спілкування між браузером користувача та сервером. Базовий приклад підключення та комунікації зображено на рисунку 4.4.

```
const socket = new WebSocket("wss://signaling-server");

socket.onopen = (e) => {
    // обробка підключення
    socket.send("call"); // запит на звінок
};

socket.onmessage = (e) => {
  // обробки повідомлень від серверу сигналінгу
};
```

Рисунок 4.4 – Лістинг прикладу коду для роботи з WS

4.1.2 STUN/TURN сервер

В офіційних нотатках, щодо початку роботи з цією технологією, у більшості випадків для того, щоб подібні програми працювали, знадобиться особливий тип

сервера, який відповідає за передачу трафіку між учасниками, тому що прямий сокет часто використовувати неможливо між клієнтами, якщо вони не знаходяться в одній локальній мережі. Такий сервер має назву TURN, що означає Traversal Using Relay NAT і є протоколом для ретрансляції мережевого трафіку [22].

Оскільки знайти безкоштовний сервер TURN досить важко, в кінцевому підсумку впровадили власний сервер STUN.

Потрібно також виконати наступні вимоги:

— сервер з встановленим Ubuntu 18.04;

— потрібно знати публічну адресу серверу;

— потрібно мати власний домен та мати доступ до DNS менеджеру, так як буде створено два сабдомени;

— SSL сертифікати для сабдомену. без безпечного протоколу впровадження вашого сервера не буде завершено, а після використання його у ваших проєктах webrtc із https воно не працюватиме;

— потрібно впевнитись, що відкриті порти 5349 та 3478;

— якщо ви використовуєте NAT, наприклад, з екземпляром Amazon EC2, обов’язково налаштуйте зовнішній IP у файлі конфігурації.

Для реалізації власного сервера STUN/TURN було вирішено будувати його на основі проєкту Coturn. Coturn – це безкоштовна реалізація сервера TURN і STUN з відкритим кодом для VoIP і WebRTC [23]. Є багато нових розширених специфікацій TURN, які виходять далеко за межі оригінального документа RFC 5766. Цей проєкт використовує код rfc5766-turn-server, як стартовий і додає до нього нові розширені функції.

Для перевірки на функціональність серверів STUN/TURN є інструмент Trickle ICE, проєкт сторінки WebRTC, який тестує функціональність trickle ICE у звичайній реалізації WebRTC. Він створює PeerConnection із зазначеними ICEServers, який міститиме інформацію про нещодавно створений сервер, а потім починає збір кандидатів для сеансу з одним звуковим потоком. Коли збираються кандидати, вони відображаються в текстовому полі нижче разом із вказівкою, коли збір кандидатів завершено.

4.2 Розробка серверної частини

Виходячи з того, що роботу по синхронізації стримінгу аудіо та відео було вирішено перекласти на засоби технології WebRTC та STUN/TURN сервери, вимоги до особливостей серверної частини зменшились. Достатньо, щоб програмна платформа підтримувала роботу з сокетами, HTTP запитами, тоді можна будувати комплексну логіку в різних модулях та підтримувати розгортання майбутніх сервісів у хмарному середовищі. Робота з мережею найчастіше покривається взаємодією з системними викликами операційної системи, доступ до яких є у більшості програмних платформ та засобів мов програмування. До можливих варіантів можна віднести: Java EE, .NET Core або NodeJS. Також є і інші варіанти, проте саме ці програмні платформи підтримуються більшістю провайдерами хмарних послуг. Не має різниці, яку саме вибрати, так як вони показують приблизно однакову швидкість роботи та розгортання. Вибір залежить лише від наявності розробників на ринку, їх кваліфікації та фінансових очікувань.

З огляду на архітектуру системи та розглянуті функціональні частини, стає зрозумілим, що кожний окремий модуль представляє собою невеликий сервіс, котрий обробляє потоки даних та запити через HTTP. До таких сервісів можна віднести усі запропоновані вище, окрім сервісу по роботі з даними користувача та телеметрії. Для таких сервісів краще використати відому трьох рівневу архітектуру, через концентрацію усієї необхідної логіки у строго описаному інтерфейсі для подальшого розширення або міграції у інший програмний додаток. Особливостями такої архітектури на рівні додатку є розділення на компонентні шари, котрі відповідають за відображення даних та взаємодію з клієнтами (контролери), шар роботи з бізнес-логікою та обчисленнями (сервісний шар) та шар доступу до даних [24]. Приклад структури проєкту на платформи .NET за таким стилем зображено на рисунку 4.5.

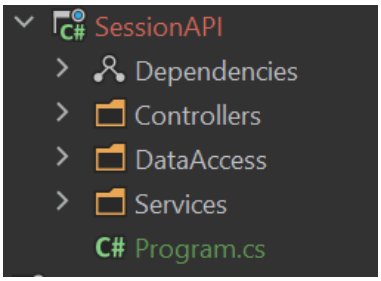


Рисунок 4.5 – Структура проєкта трьохрівневної архітектури

Розглядаючи модуль прийняття рішень на рівні сервісу роботи з сесіями та оповіщеннями, потрібно реалізувати механізм блокування паралельного доступу до функціоналу перекладу. Для таких цілей існує примітив синхронізації – монітор. Монітор – це конструкція синхронізації, яка дозволяє паралельним потокам чекати або блокувати доступ до критичної секції коду іншими потокам, та найчастіше представляє собою конструкцію lock у більшості мов програмування [25]. Також, цей модуль відповідальний за координацію одночасного перекладу тексту на різні мови, з урахуванням особливостей синхронного перекладу. Цей модуль відповідає за відслідковування статусу завершення перекладу на стороні кожного клієнта, так як присутня проблема різної довжини тексту для різних мов. Приклад реалізації описано у додатку В.

Для сервісів з комплексною логікою краще створювати продукти на базі Onion архітектури. До таких сервісів можна віднести сервіс по роботі з даними про користувачів, так як сервіс можна розширювати функціоналом підписок, слайд-презентаціями, розкладом, рекламою, питаннями до конференцій та інше. Цибулева архітектура полягає в поділі програми на шари, де кожний шар залежить лише від своїх нижніх шарів [26]. Таку назву має архітектура через те, що можна виразити схему у вигляді цибулі з серцевиною. Це дозволяє будувати комплексну логіку на рівні одного додатку, що важливо для монолітних сервісів, котрі опрацьовують більшість бізнес сутностей системи. Структуру проєкту з використанням onion архітектури зображено на рисунку 4.6.

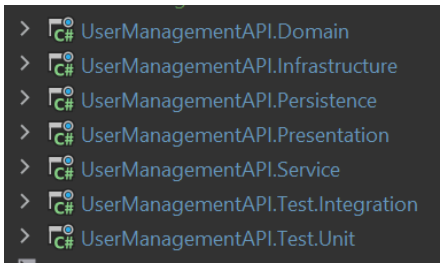


Рисунок 4.6 – Структура проєкта за onion архітектури

Сервіс по роботі зі сповіщенням користувачів, такий як сервіс оповіщень, повинен реалізувати механізми підключення по WS та мати можливість підтримувати SSE , для розширення підтримки роботи з різними браузерами. Робота з сокетами може вимагати додаткового налаштування політик доступу до мережі на хмарному провайдері, де буде розгорнутий сервіс. Так як робота з цим протоколом імплементована засобами операційних систем, то можна використовувати будь-яку програмну платформу та мову програмування.

4.3 Результати тестування

Розробивши прототип системи, було проведено декілька дослідження по роботі з перекладом. Дослідження проводились з урахуванням специфіки роботи з різними мовами, проте точність перекладу не розглядалась, так як це залежить від конкретних реалізацій сервісів по розпізнаванню та перекладу, що не є ціллю розробки у рамках даної роботи.

Було виявлено проблему, під час реалізації, зі забезпеченням прийняття рішення про завершення блокування на переклад, так як специфіка різних мов полягає у тому, що однакове за змістом речення, має різну довжину у перекладі, та потребує різного часу на озвучування.

Для вирішення цією проблеми є декілька підходів:

— розрахувати час найшвидшого або найповільнішого перекладу і збільшити або зменшити на деякий параметр швидкість озвучування отриманого перекладу;

— дозволити озвучувати кожний переклад окремо, та консолідувати момент закінчення кожного перекладу, для розблокування наступного запису.

Перший процес можна представити у форматі формули, де t_n - це час необхідний для озвучування перекладу однією мовою, x_n – це коефіцієнт на котрий потрібно зменшити результуючий час перекладу.

$$1 = \frac{(x_1 t_1 + x_2 t_2 + \cdots + x_n t_n)}{t_{min}}$$

Цей варіант, під час дослідження на різних тестових даних, показав свою недоцільність, через те що речення на деяких мовах, таких як німецька та скандинавські мови, можуть бути доволі різної довжини, що в деяких випадках дає високий коефіцієнт прискорення, у разі котрого сприймання інформації стає складним. Тестування було проведено мануальним зачитуванням одного і того самого тексту і перекладалось на різні 4 мови. Результати зображено на рисунку 4.7.

```
Вхідний текст:
============================================================

It's an ugly day today.
I know. I think it may rain.
It's the middle of summer, it shouldn't rain today.
That would be weird.
Yeah, especially since it's ninety degrees outside.
I know, it would be horrible if it rained and it was hot outside.
Yes, it would be.
I really wish it wasn't so hot every day.
Me too. I can't wait until winter.
I like winter too, but sometimes it gets too cold.
I'd rather be cold than hot.
Me too.
============================================================
Час озвучування мовою оригіналу: 00:27
Час озвучування мовою UA: 00:30
Час озвучування мовою RU: 00:31
Час озвучування мовою DE: 00:36
============================================================
```

Рисунок 4.7 – Тестування часу озвучування різних перекладів

Тому реалізовано інший підхід коли модуль прийняття рішень блокує функціонал перекладу, до того часу поки кожний клієнт не відправить сповіщення про завершення озвучування.

Іншим етапом тестування було аналіз метрик зв’язаних з часом проходження того чи іншого етапу бізнес процесу перекладу. Було проаналізовано співвідношення кількості слів у озвученому тексті до точності перекладу у відсотках представленого співвідношенням всієї кількості слів до вірно перекладених, часу від завершення запису на стороні клієнта до часу отримання відповіді іншими учасниками перекладеного аудіотреку та час необхідний на розпізнавання тексту у аудіотреці. Результати зображено у таблиці 4.1.

Таблиця 4.1 — Результати дослідження залежності довжини до критеріїв

Довжина тексту (в словах)	Час розпізнавання тексту, с	Час перекладу, мс	Час озвучування, с
2	2.1	138	1.5
10	4.5	109	3.3
20	11	175	10
50	30	263	27
100	41	281	37

Дослідження було проведено на парі мов українська-англійська. Так як, у повному процесі приймають участь сервіси розпізнавання перекладу, то отриманий результат у середньому 72%. Точність залежить від довжини тексту та складності промовлених слів, а також від мови користувача.

Висновки до розділу

У даному розділі було оглянуто підходи та необхідний програмний інтерфейс для реалізації компонентів системи на стороні клієнта та стороні серверів.

Було розроблено та проведено тестування прототипу системи. На основі прототипу було проведено дослідження залежності кількості слів у тексті, отриманому під час мовлення одного із учасників конференції, до критеріїв оцінки системи, таких як: точність, час розпізнавання мови та швидкість перекладу.

Виходячи з результатів можна зробити висновок, що при збільшені кількості тексту необхідного на переклад, наявна нелінійна необхідність збільшення часу на розпізнавання. Тексти, схожі за розміром, потребують приблизно однаковий час на розпізнавання. Проте, чим більший текст, тим зменшується точність перекладу, тому слід розмовляти більш короткими фразами, що є більш схожим до комунікації у реальному світі.

5 РОЗРОБКА СТАРТАП-ПРОЄКТУ

5.1 Опис ідеї проєкту

У цьому розділі будуть розглянуті можливості для виходу на ринок програмного забезпечення для відео-комунікації з можливостями розпізнавання мови, переваги і недоліки у порівнянні з вже існуючими на ринку рішеннями, що є аналогами.

Основною цільовою аудиторію є юридичні та фізичні особи, що мають необхідність спілкуватись з іноземцями на зручній для них мові. Також, орієнтується на навчальні програми та заклади.

Проєкт, за своєю сутністю є набором застосунків, що у комплексі надає користувачу можливість створювати та керувати конференціями з функцією перекладу під час сесії.

Основною вигодою від користування даним проєктом, є зменшення і оптимізація витрат на комунікацію з спеціалістами по всьому світі, шляхом можливості спілкування без знання мови учасників.

Представимо наведену вище інформацію у вигляді таблиці 5.1 для наглядності.

Таблиця 5.1 — Опис ідеї стартап-проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Програмне забезпечення розроблене в рамках цієї роботи є комплексним рішенням і може бути використане: - Для навчання іноземним мовам;	- логістичні компанії; - освітні заклади; - особисті цілі; - пошук однодумців.	- переклад багатьма мовами; - відображення у вигляді субтитрів з можливістю збереження

Продовження таблиці 5.1

– спілкування з людьми з різних куточків світу; – проведення рекламних заходів та конференцій.		– озвучування голосу залежно від статі учасника

В якості подальших кроків по розвитку проєкту, необхідно буде звернути увагу на інтеграцію зі сторонніми сервісами, зменшення часу затримки розпізнавання, інтеграції з корпоративними засобами комунікації, CRM системами, тощо. Наступним етапом розглянемо техніко-економічні характеристики ідеї і представимо їх у вигляді таблиці 5.2.

Таблиця 5.2 — Огляд техніко-економічних характеристик ідеї

№	Техніко-економічні характеристики ідеї	Продукція конкурентів				W	N	S
		Мій проєкт	Skype	Zoom	TimeKettle M2			
1	Оптимізація	+	-	+	+			+
2	Підтримка великої кількості учасників	-	-	+	-	+		
3	Незалежність від встановлення на персональний пристрій	+	-	+	-			+
4	Підтримку субтитрів	+	+	+	-		+	

Продовження таблиці 5.2

5	Безкоштовне використання	+	+	+	-		+	
6	Розпізнавання мови	+	-	-	+			+
7	Система для озвучування тексту	-	-	-	+		+	

З таблиці 5.2, можна зробити висновок про те, що розроблюваний продукт має ряд переваг перед деякими конкурентами, наприклад наявність можливості розпізнавання мови, незалежність від встановлення на цільову платформу та оптимізацію. Слід зазначити, що великим недоліком є підтримка невеликої кількості одночасно присутніх учасників в рамках однієї конференції, але це може бути виправлено в майбутньому. Також, великим плюсом в подальшому буде розроблення рішення для автоматичного розпізнавання пуз для початку трансляції. На даний момент розробити таке рішення неможливо, через брак необхідних даних для навчання власної математичної моделі.

5.2 Технологічний аудит ідеї проєкту

Розглянемо технічні методи для технічної реалізації ідей проєкту у вигляді таблиці 5.3.

Таблиця 5.3 — Технологічна здійсненність ідеї проєкту

№	Ідея проєкту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Клієнт конференції	JavaScript, WebRTC, інструменти роботи з Web Sockets	Наявні	Безкоштовно

Продовження таблиці 5.3

2	Розпізнавання тексту	На вибір AssemblyAI, Amazon Trascribe, Google Speech-To-Text	Наявні	Платно
3	Переклад тексту	На вибір Google Tranlate API, Microsoft Translation API	Наявні	Є безкоштовний план для розробки, платно для комераційного використання
4	Озвучування тексту	.NET Core, Node.JS, Google Cloud Text-to-Speech	Наявні	Безкоштовно при розробці, платно для комераційного використання
5	Нотифікування клієнтів	На вибір SignalR, SocketIO	Наявні	Безкоштовно при розробці, платно для комераційного використання
6	Серверна частина	На вибір .NET Core, Java EE, Node.js	Наявні	Безкоштовно при розробці, платне розгортування для комерційного використання
7	Клієнтська частина	ReactJS або Angular	Наявні	Безкоштовно

Можна зробити висновок про те, що усі необхідні технології реалізації проєкту відео-конференцій з розпізнаванням мови засобів наявні, і мають безкоштовний план для використання при розробці. Проте, практично за усі доведеться платити при комерційному розгортанні у хмарі.

5.3 Аналіз ринкових можливостей запуску стартап-проєкту

Таблиця 5.4 — Попередня характеристика потенційного ринку

№	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	4
2	Загальний обсяг продаж, грн./ум.од	Немає наявних даних у відкритих джерелах інформації
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу	Відсутні
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі або по ринку, %	Немає наявних даних у відкритих джерелах інформації

Таблиця 5.5 — Характеристика потенційних клієнтів стартап-проєкту

№	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці цільових груп клієнтів	Вимоги споживачів до товару

Продовження таблиці 5.5

1	Підтримка багатьох мов Швидкість перекладу Можливість для інтеграції Стабільність Простота використання	Логістичні компанії Міжнародні компанії Освітні заклади Фізичні особи	Ціль використання системи Різна платеспроміжність Знання іноземних мов	Легкість у використанні Низький поріг входження Зручний графічний інтерфейс

Висновок: враховуючи кількість головних гравців по ринку, зростаючу динаміку ринку, невелику кількість конкурентів та середню норму рентабельності можна зробити висновок, що на даний момент, ринок для входження стартап-продукту є привабливим.

Таблиця 5.6 — Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Малоефективна рекламна кампанія	Погане інформування цільової аудиторії через низькоякісну рекламу	Удосконалення рекламної кампанії; Найм більш компетентного маркетолога.
2	Складність використання системи	Нові користувачі бояться пробувати нове через складність і заплутаність системи.	Збір відгуків від клієнтів з метою спрощення; Створення форуму і вікі сайту.

Продовження таблиці 5.6

3	Великі компанії конкуренти	Наявність конкурентів із значними ресурсами, котрі надають схожі рішення	Зменшення ціни на поставлену послугу; Розробка унікальних характеристик товару.
4	Кошти на розробку та підтримку продукту	Закінчення грошей та недостатнє фінансування	Залучення додаткових інвесторів; Ітеративна розробка продукту
5	Поява повного аналогу	Вихід аналогу даного товару може призвести до знецінення та безідейності даного товару	Вихід товару на ринок в коротші строки з не повною, але достатньою, функціональністю для зацікавлення усіх цільових аудиторій; Агресивна рекламна кампанія.

Таблиця 5.7 — Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Новий продукт	Вихід на ринок; Надання нових рішень у сфері	Розробка нової функціональності; Вихід нової продукції на ринок; Надання різноманітних типів підписок в залежності від потреб користувача.

Продовження таблиці 5.7

2	Вихід аналогу	Надати продукт з більш багатим функціоналом, за ті ж кошти	Постійний аналіз ринку та відгуків користувачів задля задоволення їх потреб та надання функціональності у найкоротші строки; Програми лояльності для постійних клієнтів.
3	Зворотній зв’язок від користувачів	Можливість отримання необхідної інформації для вдосконалення продукту	Збір відгуків користувачів прямо у веб-інтерфейсі та реакція на них з боку команди розробників.
4	Реферальна програма	При достатньому попиту на програмне забезпечення, надавати бонуси клієнтам	Залучення додаткових користувачів і їх коштів

Таблиця 5.8 — Ступеневий аналіз конкуренції на ринку

№	Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1	Тип конкуренції: чиста	На ринку є умови для входу та виходу; Ціна корелює між суперниками; Аналоги схожі по функціональності.	Розробка продукту з характеристиками, які покривають сфери вживання що не покривають інші товари-замінники; Кореляція цін до товарів замінників;

Продовження таблиці 5.8

2	Рівень конкурентної боротьби: світовий	Практично усі рішення-аналоги не мають прив'язки до певної країни, хоч часто основні клієнти походять із країн СНД.	Вихід на ринок продукту з необхідною функціональністю; Налагодження маркетингу на основних Інтернет ресурсах задля охоплення великої кількості користувачів.
3	Галузева ознака: Міжгалузева	Конкуренція спостерігається в усіх сферах, де використовуються комунікація між людьми.	Продукт вирішує специфічні задачі; Створення нового функціоналу; Наявність документації та он-лайн підтримки.
4	Конкуренція за видами товарів: товарно-видова	Конкуренція між товарами різного виду, що можуть виконувати схожі функції	Створення нового функціоналу; Надання підтримки.
5	Характер конкурентних переваг: цінова та не цінова	Конкуренція проводиться за рахунок зниження вартості та збільшенні функціоналу.	Покращення продукту з одночасним зниженням вартості; Надання різних типів підписок в залежності від потреб клієнта.
6	За інтенсивністю: марочна	Бренд відіграє важливу роль, клієнти довіряють знайомим брендам.	Побудова власного обличчя компанії; Грамотне ведення соціальних мереж;Представлення в медіа.

Таблиця 5.9 — Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Присутні	Монополія	Розміри поставок	Зменшення цін	Ціна
Висновки	Прямих конкурентів багато, завдяки цьому конкурента боротьба є інтенсивною.	Потенційні конкуренти вже присутні на ринок, за рахунок цього конкуренція може сильно вирости.	Невеликі обсяги закупки товарів.	Через велику пропозицію клієнти диктують умови.	Ціна не має бути дорожчою ніж ціна прямих конкурентів.

З огляду на таблиці вище, можна зробити висновок, що не дивлячись на доволі сильну конкуренцію на ринку, кожен з продуктів має своє особливості що дозволяє зайняти певну нішу. Таким чином, якщо надати загальні рішенні, а з часом, певну специфічну функціональність то вийти на цей ринок є доволі здійсненною і можливою задачею.

Для успішного виконання задачі по виходу на ринок, окрім функціональності, необхідно забезпечити наявність різних і гнучких планів підписок, кваліфіковану підтримку для залучення якомога більше клієнтів і їх втримання.

Таблиця 5.10 — Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування
1	Доступна ціна	Товар доступний для різних клієнтів.
2	Точність перекладу	Покращує користувацький досвід
3	Швидкість перекладу	Позитивно впливає на сприйняття товару потенційними клієнтами
4	Стабільність	Покращення досвіду роботи з системою.
5	Робота у реальному часі	Робота у реальному часі дозволяє відобразити спілкування як у житті

Таблиця 5.11 — Порівняльний аналіз сильних та слабких сторін ПЗ

№	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з запропонованим						
			-3	-2	-1	0	+1	+2	+3
1	Доступна ціна	18				+			
2	Точність перекладу	13		+			+		
3	Швидкість перекладу	20		+		+	+		

Продовження таблиці 5.11

4	Стабільність	16		+	+	+			
5	Робота у реальному часі	16		+	+	+			
6	Доступна ціна	18		+	+	+			

Таблиця 5.12 — SWOT аналіз стартап-проєкту

Сильні сторони (S):	Слабкі сторони (W):
доступна ціна легкість використання швидкість перекладу стабільність роботи робота в реальному часі робота з будь-якого куточка світу	наявність конкурентів підтримка малої кількості учасників затримка по завершенню зчитування тексту невідомий бренд
Можливості (O):	Загрози (T):
збільшений попит використання механізмів машинного навчання	знижений інтерес потенційних клієнтів складність входження на ринок

З аналізу сильних та слабких сторін проєкту, необхідно сформулювати альтернативні шляхи ринкової поведінки, ймовірність реалізації даних шляхів, та строки їх реалізації. Перелік наведено в таблиці 5.13.

Зазвичай, природа стартапу полягає в реалізації проєкту невеликою кількістю людей, через наявні обмеження ресурсів. Найчастіше, це до п’яти людей у команді.

Таблиця 5.13 — Альтернативи ринкового впровадження стартап-проєкту

№	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Система для навчання онлайн курсам	Необхідно залучити інвесторів для додаткових матеріальних ресурсів	6-12 місяців
2	Платформа для об’єднання перекладачів для вирішення приватних замовлень	Необхідно залучити інвесторів для додаткових матеріальних ресурсів	8 місяців
3	Платформа для роботи з голосовими командами	Необхідно залучити інвесторів для додаткових матеріальних ресурсів	6 місяців

5.4 Розроблення ринкової стратегії проєкту

Таблиця 5.14 — Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Міжнародні підприємства	Висока, оскільки компанії зацікавлені у більш ефективному використанні наявних ресурсів	80% - високий попит	Висока конкуренція	Легко, оскільки компанії мають досить ресурсів для впровадження.
2	Освітні заклади	Висока, оскільки компанії зацікавлені у більш ефективному використанні наявних ресурсів	70% - середній попит	Висока конкуренція	Легко, оскільки компанії мають досить ресурсів для впровадження.

Продовження таблиці 5.14

3	Фізичні особи	Середня, адже не кожному необхідна ефективність	50% - середній попит	Низька конкуренція	З труднощами, оскільки ціна може бути завеликою
	Які цільові групи обрано: міжнародні підприємства, освітні заклади та фізичні особи.				

Згідно даним, що наведені у таблиці вище, найбільш підходящими цільовими групами є компанії і підприємства, що у процесі діяльності багато мають комунікацій з спеціалістами з різних куточків світу. Таким чином, пропонується зробити основний фокус на міжнародні компанії та освітні заклади. Наступним етапом, визначим базову стратегію розвитку, дані наведені у таблиці 5.15.

Таблиця 5.15 — Визначення базової стратегії розвитку

Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Система для навчання онлайн курсам	Інтенсивний розподіл	Зниження ступеню замінності товару; Прихильність клієнтів; Відмітні властивості товару; Відмітні характеристики товару;	Стратегія диференціації

Продовження таблиці 5.15

Платформа для об’єднання перекладачів для вирішення приватних замовлень	Інтенсивний розподіл	Зручність та легкість у користуванні, не потребує додаткових налаштування	Стратегія диференціації
Платформа для роботи з голосовими командами	Ексклюзивний розподіл	Зручність, гнучкість та легкість у використанні. Широка можливість кастомізацій.	Стратегія диференціації

Таблиця 5.16 — Визначення базової стратегії конкурентної поведінки

Чи є проєкт «першопрохідцем» на ринку	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, які?	Стратегія конкурентної поведінки
Ні, оскільки є товари-замінники, але дані товари замінники не мають деякого необхідного функціоналу	Так, ціль компанії знайти нових споживачів та, частково, забрати існуючих у конкурентів	Компанія частково копіює характеристики товару конкурента. Основна ціль компанії розробка нового унікального функціоналу, з підтримкою основного	Стратегія заняття конкурентної ніші

Таблиця 5.17 — Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проєкту	Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту
1	Веб-додаток	Стратегія диференціації	Можна не розгортати на кожній станції; Швидкість роботи; Підтримка багатьох мов.	Кроссплатформенність; Швидкий вхід в систему.
2	Легкість у використанні системою	Стратегія диференціації	Зручний графічний інтерфейс	Інтуїтивно зрозумілий інтерфейс; Візуалізація процесу роботи з перекладом та трансляцією

Згідно наведених даних, в якості базової стратегії розвитку, компанією обирається стратегія диференціації, а в якості базової стратегії конкурентної поведінки стратегію заняття певної конкурентної ніші.

5.5 Розроблення маркетингової програми стартап-проєкту

Таблиця 5.18 — Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Зручний графічний інтерфейс	Сучасний графічний, адаптований інтерфейс, який швидко працює.	Сучасна архітектура; приємний інтерфейс; швидкість і стабільність.
2	Швидкий переклад аудіо-потоку	Набагато швидше аналізує та розпізнає текст у вхідному аудіо-каналі	Використання актуальних бібліотек для розробки функціоналу. Безпека даних.
3	Переклад різними мовами	Зручний спосіб відображення перекладених даних	Використання актуальних технологій.

Таблиця 5.19 — Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
1. Товар за задумом	програмне забезпечення для відео-комунікації з можливостями розпізнавання мови для перекладу

Продовження таблиці 5.19

2. Товар із підкріпленням	До продажу: наявна повна документація, наявний пробний період, можливість організовувати конференції та рекламні акції
	Після продажу: додаткова підтримка спеціалістів налаштування, підтримка з боку розробника
За рахунок чого потенційний товар буде захищено від копіювання: за рахунок патенту і авторських прав	

Таблиця 5.20 — Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
Немає	Оскільки підписка зазвичай купується для користувача, то ціна індивідуальна для компанії. В середньому від 300 до 1000 грн на місяць.	Не менш як 20тис грн на місяць	200 – 1.5 тис. гривень на місяць

Таблиця 5.21 — Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Клієнти купують продукт лише у компанії-розробника	Розробка продукту; Реклама; Підтримка.	Канал нульового рівня (виробник безпосередньо продає товар клієнту)	Прямий збут

Таблиця 5.22 — Концепція маркетингових комунікацій

№	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Клієнти дізнаються про товар з реклами в інтернеті	Контекстна реклама; Реклама в соцмережах;	Швидкий, безкрайній, зручний, вирішує проблему комунікації	Інформування про існування продукту та його переваги	Використання технологій для вирішення мовленнєвих бар’єрів

Було описано маркетингову програму, визначено ключові переваги в концепції потенційного товару, описана його модель, визначені ціни, системи збуту, концепція маркетингових комунікацій.

Висновки по розділу

У даному розділі дисертації був описаний потенціальний стартап-проєкт, що представляє собою продаж програмного забезпечення розробленого в рамках даної роботи. Були розглянуті: рентабельність роботи, попит, динаміку ринку, потенційні цільові групи. Можна зробити висновок, що проєкт має право на існування і є перспективним, хоч і існують певні перепони на шляху входження в ринкову нішу: високорозвинені рішення від конкурентів.

Також, було запропоновано три альтернативи для подальшого розвитку проєкту: система для навчання онлайн курсам, платформа для об’єднання перекладачів для вирішення приватних замовлень та платформа роботи з голосовими командами.

Таким чином, впровадження стартап-проєкту і подальша розробка в рамках цього проєкту є доцільним.

ВИСНОВКИ

У роботі розглянуто проблему міжмовної комунікації та засоби її реалізації. Виявлено потребу в удосконаленні підходів до розробки систем відеокомунікації з розширеннями для вузьких галузей, використовуючи засоби загального використання.

Основною метою була розробка архітектурних рішень та методів для побудови системи відеокомунікації зі спеціалізацією на розпізнаванні мови під час сеансу конференції для перекладу іноземною мовою з можливістю подальшого озвучення або збереження у цифровому вигляді.

В рамках роботи над дисертацією було розглянуто предметну область, проведено аналіз існуючих рішень, таких як: Zoom, Skype, Google Translate та гарнітури-перекладачі, було виявлено суттєві елементи системи, а саме: принципи роботи протоколів по стримінгу аудіо- та відеосигналів, методи та засоби зчитування людської мови з браузера клієнта, хмарні сервіси по роботі з перекладом тексту та інструменти для перетворення результатів у аудіопотоки.

Користуючись результатами аналізу функціональних та нефункціональних вимог та описом критичних для системи бізнес-сценаріїв було розроблено архітектуру систему. Відповідно до розробленої архітектури було проаналізовано та порівняно методи та технології реалізації, до яких відносяться архітектурні рішення, протоколи комунікації, сервіси обробки аудіоканалів, бази даних та інфраструктурні провайдери.

Були описані аспекти, необхідні для реалізації програмних інтерфейсів частин системи, та реалізовано прототип. На базі прототипу було проведено дослідження залежностей кількості слів у вхідному аудіотреку до точності, швидкості розпізнавання та часу перекладу.

В якості обґрунтування економічної доцільності подальшої роботи над проєктом розроблено стартап-проєкт, в рамках якого описано ідею проєкту, проведено технологічний аудит ідеї і аналіз ринкових можливостей, розроблена маркетингова стратегія.

Практична значимість одержаних результатів полягає в тому, що розроблена архітектура організації відеоконференцій із можливістю автоматичного перекладу мови співрозмовника та має широкі можливості для розвитку і подальшого покращення. Запропонована система може зацікавити широке коло користувачів.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Skype Official Website [Електронний ресурс] – Режим доступу до ресурсу: https://www.skype.com.

2) Zoom Meetings [Електронний ресурс] – Режим доступу до ресурсу: https://explore.zoom.us/en/products/meetings.

3) Google Translation API [Електронний ресурс] – Режим доступу до ресурсу: https://cloud.google.com/translate.

4) TimeKettle Official Website [Електронний ресурс] – Режим доступу до ресурсу: https://www.timekettle.co.

5) Monolithic vs. Microservices: a guide to application architecture [Електронний ресурс] – Режим доступу до ресурсу: https://www.talend.com/resources/monolithic-architecture.

6) Ingeno Joseph. Software Architect's Handbook. / Ingeno, Joseph. - Packt Publishing, 2018. 175-176 с.

7) What is DevOps? [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/devops/what-is-devops.

8) Introduction to the Real-time Transport Protocol [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/Intro_to_RTP.

9) Streaming Protocols: Everything You Need to Know [Електронний ресурс] – Режим доступу до ресурсу: https://www.wowza.com/blog/streaming-protocols.

10) Salvatore Loreto. Real-Time Communication with WebRTC. / Loreto, Salvatore. - O'Reilly Media, 2014. 59-62 с.

11) Santos-Gonzalez I., Rivero-Garcia A., Molina-Gil J., Caballero-Gil P. Implementation and Analysis of Real-Time Streaming Protocols. – MDPI, 2017.

12) Dierks T., Rescorla E. The Transport Layer Security (TLS) Protocol. – Internet Engineering Task Force (IETF), 2008.

13) Thornburgh M. Adobe's Secure Real-Time Media Flow Protocol. – Internet Engineering Task Force (IETF), 2013.

14) Web Speech API [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Web/API/Web_Speech_API.

15) CAP Theorem [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/swlh/cap-theorem-in-distributed-systems-edd967e7bdf4

16) Speech-to-Text transcript accuracy rate among leading companies worldwide in 2020 [Електронний ресурс] – Режим доступу до ресурсу: https://www.statista.com/statistics/1133833/speech-to-text-transcript-accuracy-rate-among-leading-companies.

17) How Accurate is Google Translate [Електронний ресурс] – Режим доступу до ресурсу: https://thelanguagedoctors.org/how-accurate-is-google-translate.

18) Alshafie Gafaar: SQL vs. NoSQL / Alshafie Gafaar, Saife Eldin // Journal of Multidisplinary Engineering Sciense Studies Vol 3 -. 2017. - №5. - 1790-1791c.

19) AWS vs AZURE [Електронний ресурс] – Режим доступу до ресурсу: https://www.educba.com/aws-vs-azure.

20) WebRTC overview [Електронний ресурс] – Режим доступу до ресурсу: www.webrtc.org.

21) WebRTC Adapter file [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/webrtc/adapter

22) What are STUN, TURN, and ICE? [Електронний ресурс] – Режим доступу до ресурсу: https://developer.liveswitch.io/liveswitch-server/guides/what-are-stun-turn-and-ice.html.

23) Coturn Main Page [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/coturn/coturn.

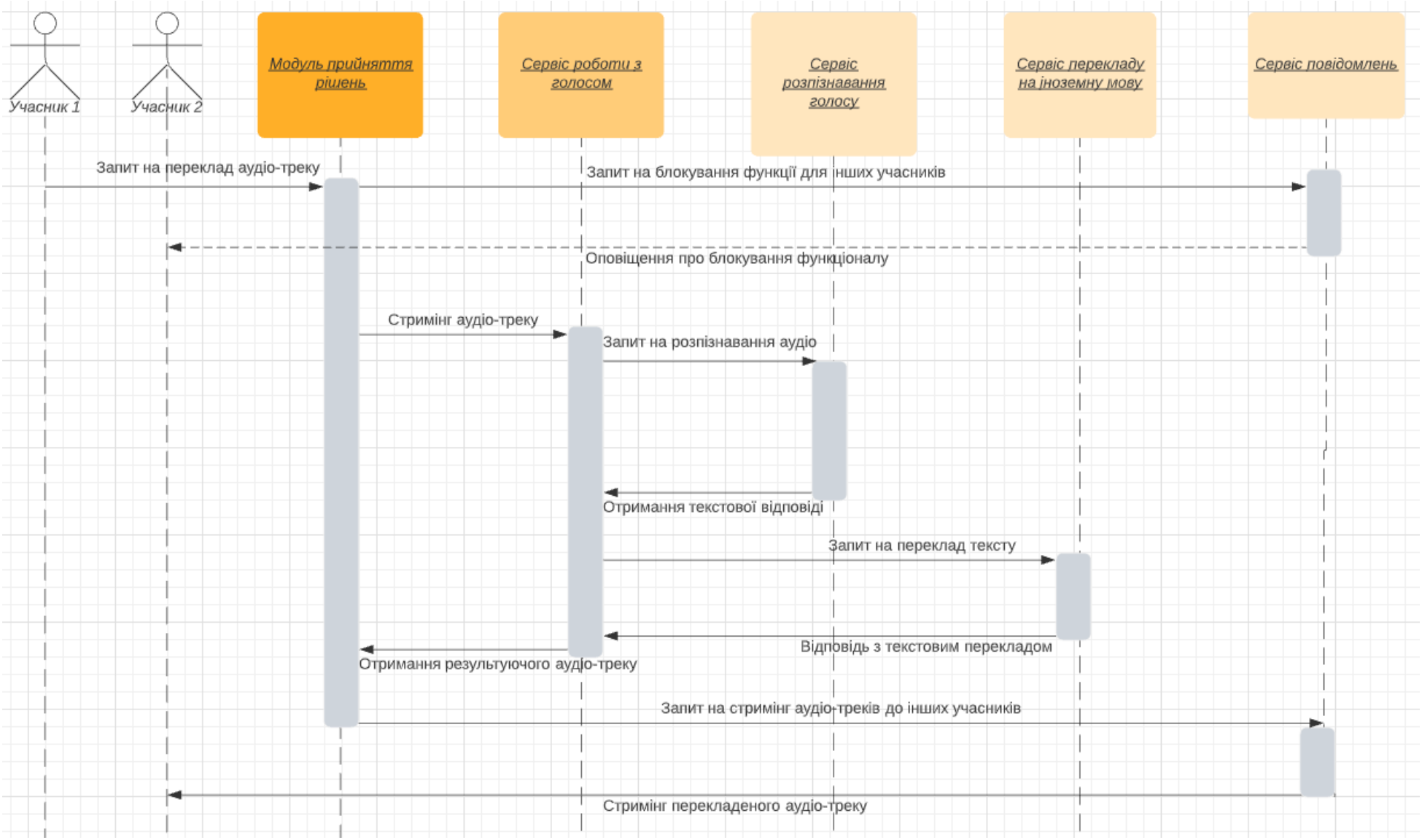
24) Monitors in Process Synchronization [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/monitors-in-process-synchronization.

25) Three-Tier Architecture [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/cloud/learn/three-tier-architecture.

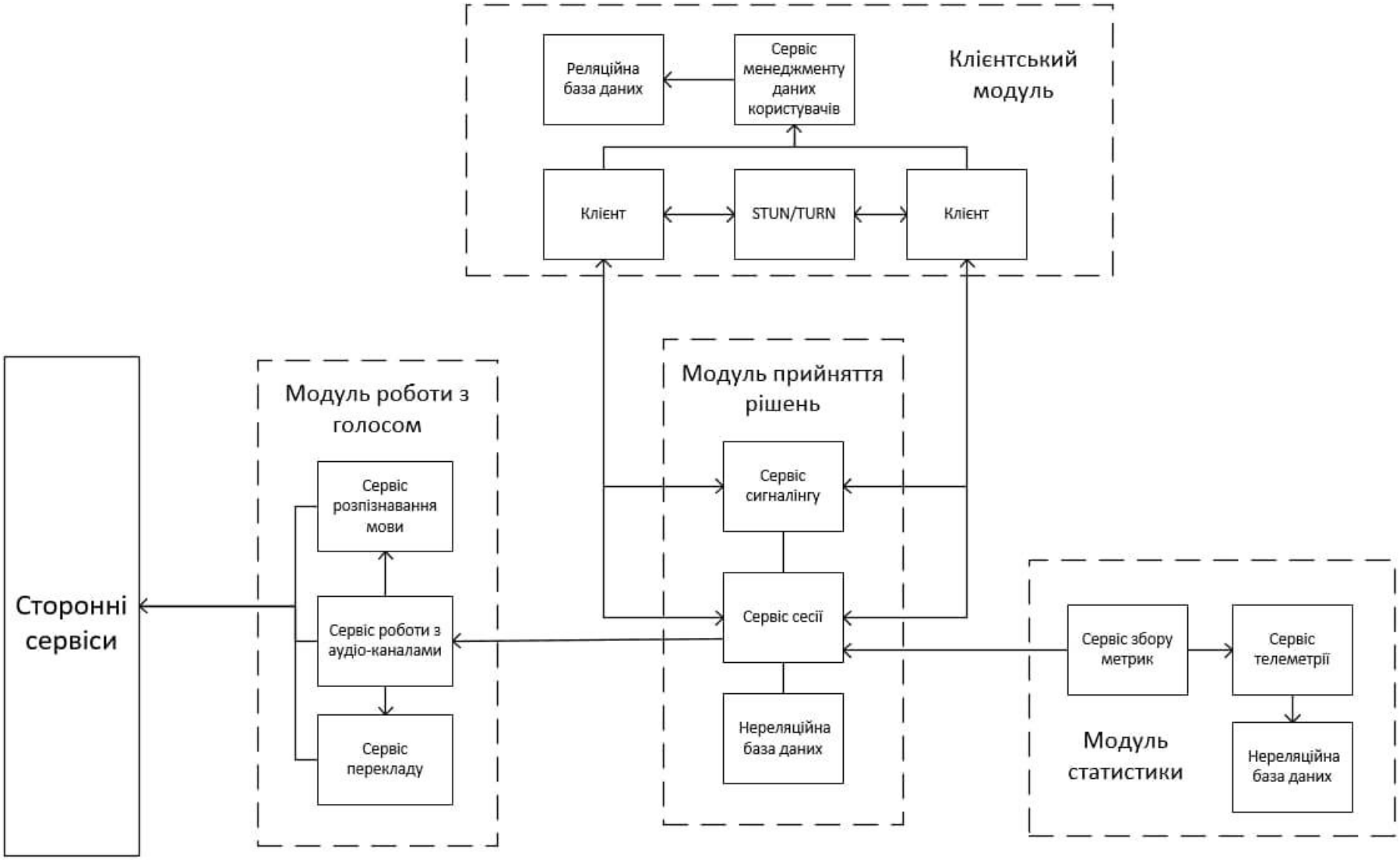
26) Understanding Onion Architecture [Електронний ресурс] – Режим доступу до ресурсу: https://www.codeguru.com/csharp/understanding-onion-architecture.

ДОДАТКИ

ДОДАТОК А – ДІАГРАМА ПОСЛІДОВНОСТІ ПРОЦЕСУ ПЕРЕКЛАДУ ГОЛОСУ УЧАСНИКА



ДОДАТОК Б – АХІТЕКТУРА СИСТЕМИ



ДОДАТОК В – ЛІСТИНГ КОДУ ПРОТОТИПУ

Лістинг В.1 - Клієнтська частина по роботі з перекладом та озвучуванням

```
var final_transcript = "";
var recognizing = false;
var ignore_onend;
var start_timestamp;

if (!("webkitSpeechRecognition" in window)) {
  upgrade();
} else {
  start_button.style.display = "inline-block";
  var recognition = new webkitSpeechRecognition();
  recognition.continuous = true;
  recognition.interimResults = true;

  recognition.onstart = function () {
    recognizing = true;
    showInfo("info_speak_now");
    start_img.src = "assets/mic-animate.gif";
  };

  recognition.onerror = function (event) {
    if (event.error == "no-speech") {
      start_img.src = "assets/mic.gif";
      showInfo("info_no_speech");
      ignore_onend = true;
    }
    if (event.error == "audio-capture") {
      start_img.src = "assets/mic.gif";
      showInfo("info_no_microphone");
      ignore_onend = true;
    }
    if (event.error == "not-allowed") {
      if (event.timeStamp - start_timestamp < 100) {
        showInfo("info_blocked");
      } else {
        showInfo("info_denied");
      }
      ignore_onend = true;
    }
  };

  recognition.onend = function () {
    recognizing = false;
    if (ignore_onend) {
      return;
    }
    start_img.src = "assets/mic.gif";
    if (!final_transcript) {
      showInfo("info_start");
      return;
    }
    showInfo("");

    if (window.getSelection) {
      window.getSelection().removeAllRanges();
      var range = document.createRange();
      range.selectNode(document.getElementById("final_span"));
      window.getSelection().addRange(range);
```

```
    }

    const targetLang = langs[select_language.value][1];
    const [targetLangCode, _] = targetLang.toLowerCase().split("_");

    translate(final_transcript, "en", targetLangCode)
      .then((res) => {
        const elem = document.getElementById("translation");
        if (elem) {
          elem.innerHTML = res;
        }

        const accessToken = document.getElementById("token")?.value || "";

        return convertToSpeech(res, targetLang, accessToken);
      })
      .then((audio) => audio.play());
  };

  recognition.onresult = function (event) {
    var interim_transcript = "";
    if (typeof event.results == "undefined") {
      recognition.onend = null;
      recognition.stop();
      upgrade();
      return;
    }
    let allFinalFlag = true;
    for (var i = event.resultIndex; i < event.results.length; ++i) {
      if (event.results[i].isFinal) {
        final_transcript += event.results[i][0].transcript;
      } else {
        allFinalFlag = false;
        interim_transcript += event.results[i][0].transcript;
      }
    }
    if (allFinalFlag === true) {
      recognition.stop();
    }
    final_transcript = capitalize(final_transcript);
    final_span.innerHTML = linebreak(final_transcript);
    interim_span.innerHTML = linebreak(interim_transcript);

    if (final_transcript || interim_transcript) {
      showButtons("inline-block");
    }
  };
}

function upgrade() {
  start_button.style.visibility = "hidden";
  showInfo("info_upgrade");
}

var two_line = /\n\n/g;
var one_line = /\n/g;
function linebreak(s) {
  return s.replace(two_line, "<p></p>").replace(one_line, "<br>");
}

var first_char = /\S/;
function capitalize(s) {
```

```
  return s.replace(first_char, function (m) {
    return m.toUpperCase();
  });
}

function startButton(event) {
  if (recognizing) {
    recognition.stop();
    return;
  }
  final_transcript = "";
  recognition.lang = "en-US";
  recognition.start();
  ignore_onend = false;
  final_span.innerHTML = "";
  interim_span.innerHTML = "";
  start_img.src = "assets/mic-slash.gif";
  showInfo("info_allow");
  showButtons("none");
  start_timestamp = event.timeStamp;
}

function showInfo(s) {
  if (s) {
    for (var child = info.firstChild; child; child = child.nextSibling) {
      if (child.style) {
        child.style.display = child.id == s ? "inline" : "none";
      }
    }
    info.style.visibility = "visible";
  } else {
    info.style.visibility = "hidden";
  }
}

var current_style;
function showButtons(style) {
  if (style == current_style) {
    return;
  }
  current_style = style;
}

function translate(inputText, source, target) {
  const qp = new URLSearchParams({
    q: inputText,
    source,
    target,
    format: "text",
    key: GOOGLE_CLOUD_TRANSLATE_KEY,
  });
  return fetch(
    `https://translation.googleapis.com/language/translate/v2?${qp.toString()}`
  )
    .then((res) => res.json())
    .then((res) => res?.data?.translations[0]?.translatedText);
}

function convertToSpeech(inputText, localeLang, accessToken) {
  return fetch("https://texttospeech.googleapis.com/v1beta1/text:synthesize", {
    method: "POST",
    headers: {
```

```
      Authorization: `Bearer ${accessToken}`,
    },
    body: JSON.stringify({
      input: {
        text: inputText,
      },
      voice: {
        languageCode: localeLang,
      },
      audioConfig: {
        audioEncoding: "OGG_OPUS",
      },
    }),
  })
    .then((res) => res.json())
    .then(
      (data) =>
        new Audio("data:audio/ogg; codecs=opus;base64," + data.audioContent)
    );
}
```

Лістинг В.2 - Клієнтська частина по роботі з комунікацією

```
const localVideo = document.getElementById('localVideo');
const remoteVideo = document.getElementById('remoteVideo');
const offerOptions = {
    offerToReceiveAudio: 1,
    offerToReceiveVideo: 1
  };
const stream = await navigator.mediaDevices.getUserMedia({audio: true,
video: true});
localVideo.srcObject = stream;
localStream = stream;
const localConfif = { }

local = new RTCPeerConnection(localConfif);
local.addEventListener('icecandidate', e => {
    await pc1.addIceCandidate(e.candidate);
});
pc1.addEventListener('iceconnectionstatechange', e => onIceStateChange(pc1,
e));

var socket = io();

socket.on('call', function(msg) {
    console.log('Starting call');

    participant = new RTCPeerConnection(msg.configuration);
    participant.addEventListener('icecandidate', e =>
onIceCandidate(participant, e));
    participant.addEventListener('track', gotRemoteStream);
    localStream.getTracks().forEach(track => local.addTrack(track,
localStream));
```

```
    participant.addEventListener('icecandidate', e =>
onIceCandidate(participant, e));
    participant.addEventListener('iceconnectionstatechange', e =>
onIceStateChange(participant, e));
    participant.addEventListener('track', gotRemoteStream);

    localStream.getTracks().forEach(track => local.addTrack(track,
localStream));
    try {
      const offer = await pc1.createOffer(offerOptions);
      await onCreateOfferSuccess(offer);
    } catch (e) {
      onCreateSessionDescriptionError(e);
    }
});
```

Лістинг В.3 - Серверна частина прийняття рішень

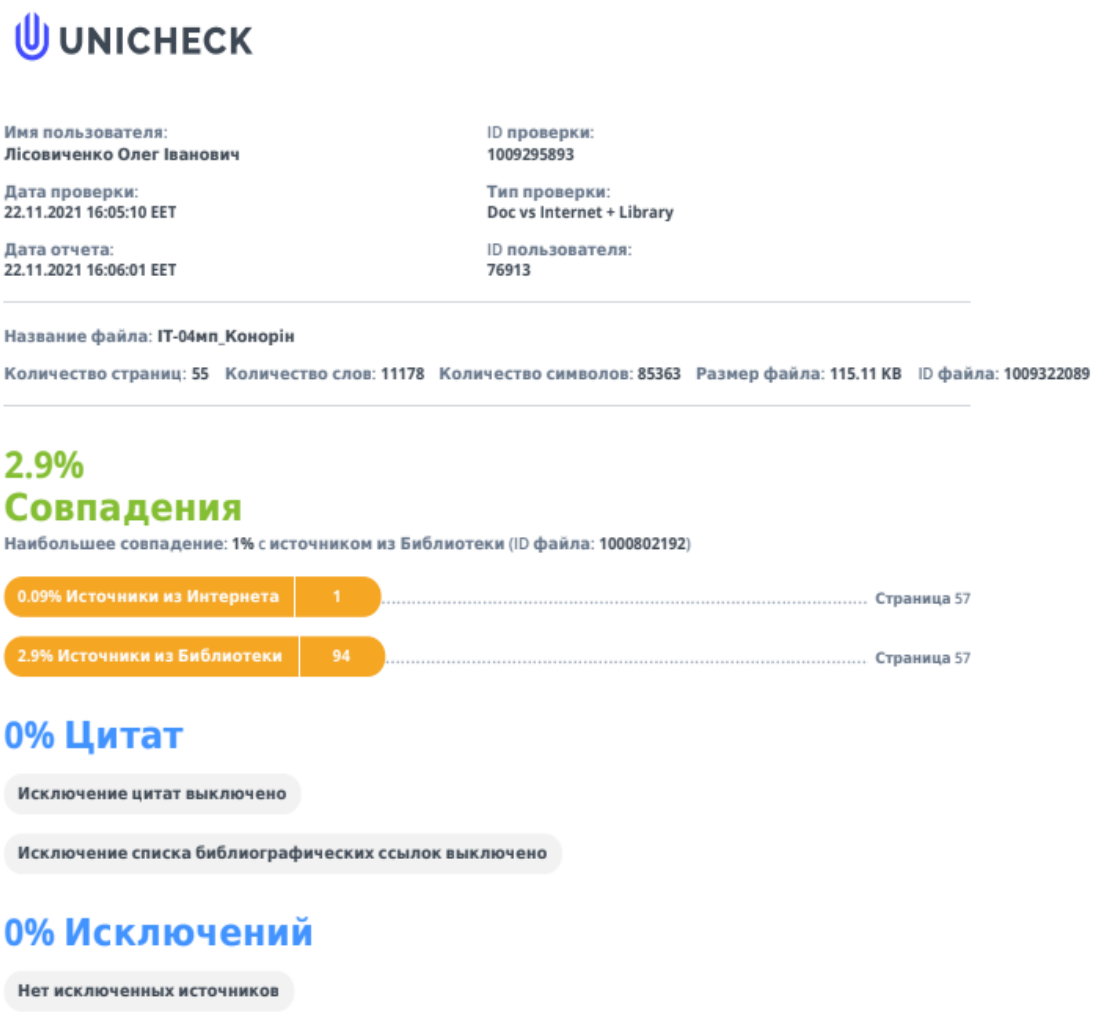
```
    public class ConferenceClient
    {
        public DependencyResolver dependencyResolver = new DependencyResolver();
        public List<SessionManager> Sessions { get; set; }

        public State state = new State();

        public Task CreateSession(ConferenceInfo info)
        {
            Sessions.Add(new SessionManager(info,
dependencyResolver.Get<NotificationClient>()));
            await state.CommitChangesAsync();
        }
    }
    public class SessionManager
    {
        private object locker = new object();
        public List<Participant> Participants { get; set; } = new List<Participant>();
        public VoiceAPI Client { get; set; }

        public SessionManager(ConferenceInfo info, NotificationClient notificationClient)
        {
            info = info ?? throw new ArgumentNullException(nameof(info));
            notificationClient.On("FinishTranslation", msg =>
            {
                var participant = Participants.FirstOrDefault(x => x.Id ==
msg.PartisipantId);
                participant.HasFinished = true;
            });
        }
        public async Task TryTranslate(Payload payload)
        {
            lock (locker)
            {
                if (!Participants.All(x => x.HasFinished))
                    return;
                Client.SendAsync(Payload)
            }
        }
    }
```

ДОДАТОК Д – РЕЗУЛЬТАТ ПЕРЕВІРКИ НА СПІВПАДІННЯ



UNICHECK

Имя пользователя:
Лісовиченко Олег Іванович

ID проверки:
1009295893

Дата проверки:
22.11.2021 16:05:10 EET

Тип проверки:
Doc vs Internet + Library

Дата отчета:
22.11.2021 16:06:01 EET

ID пользователя:
76913

Название файла: IT-04мп_Конорін

Количество страниц: 55 Количество слов: 11178 Количество символов: 85363 Размер файла: 115.11 KB ID файла: 1009322089

2.9%
Совпадения

Наибольшее совпадение: 1% с источником из Библиотеки (ID файла: 1000802192)

0.09% Источники из Интернета 1 ..... Страница 57

2.9% Источники из Библиотеки 94 ..... Страница 57

0% Цитат

Исключение цитат выключено

Исключение списка библиографических ссылок выключено

0% Исключений

Нет исключенных источников