

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
(освітня програма «Комп'ютерна графіка та розробка ігор»)
на тему:

**«Розробка сервісу для одночасного перегляду
відеоконтенту віддаленими користувачами»**

Виконав студент групи КН-19-1
ПАЛАМАРЧУК Микита Павлович

Керівник роботи:
ГРАФ Марина Сергіївна

Рецензент:
ПЕТРОСЯН Руслан Валерікович

Житомир – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних наук

Марина ГРАФ

«05» січня 2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу освітнього ступеня «бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
(освітня програма «Комп'ютерна графіка та розробка ігор»)

Здобувач вищої освіти: **ПАЛАМАРЧУК Микита Павлович**

Керівник роботи: **ГРАФ Марина Сергіївна**

Тема роботи: **«Розробка сервісу для одночасного перегляду відеоконтенту віддаленими користувачами»,**

затверджена наказом закладу вищої освіти від **«05» січня 2023 р., №10/с**

Термін здачі закінченої роботи: **«10» червня 2023 р.**

Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі. Пошук, огляд та аналіз аналогічних розробок. Формулювання технічного завдання. Опрацювання літературних джерел.	2 лютого 2023 – 5 березня 2023	Виконано
2	Проектування структури системи	6 березня 2023 – 25 березня 2023	Виконано
3	Написання програмного коду	26 березня 2023 – 16 травня 2023	Виконано
4	Тестування системи	16 травня 2023 – 1 червня 2023	Виконано
5	Оформлення пояснювальної записки	10 травня 2023 – 5 червня 2023	Виконано
6	Попередній захист	10 червня 2023	Виконано

Здобувач вищої освіти **Микита ПАЛАМАРЧУК**
Керівник **Марина ГРАФ**

РЕФЕРАТ

Пояснювальна записка до випускної кваліфікаційної роботи бакалавра складається з переліку умовних позначень, вступу, трьох розділів, висновків та списку використаної літератури, а також наявних додатків.

Пояснювальна записка до випускної роботи містить 60 сторінок, 40 ілюстрацій. Список використаних джерел містить 25 найменувань і розміщується на 3 сторінках. Загальний об'єм роботи – 81 сторінка.

У першому розділі було проведено аналіз проблематики, розглянуто сервіси-аналоги, створено концепт-документ та визначено набір інструментальних засобів розробки.

У другому розділі було спроектовано архітектуру системи та написаний програмний код.

У третьому розділі було детально описано користувацький інтерфейс розширення та взаємодія з ним. Проведено процес тестування та усунення недоліків.

Розділ висновки вміщує загальні підсумки розробки сервісу для одночасного перегляду віддаленими користувачами.

					КН.КР.Б – 122 – 13 – ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата					
Розроб.		М.П.ПАЛАМАРЧУК			Розробка сервісу для одночасного перегляду відеоконтенту віддаленими користувачами	Літ.	Арк.	Аркушів	
Керівник		М.С. ГРАФ					3	81	
Рецензент		Р.В. ПЕТРОСЯН				Житомирська політехніка, група КН-19-1			
Н. контр.									
Зав. каф.		М.С. ГРАФ							
					Пояснювальна записка				

ABSTRACT

The explanatory note to the bachelor's thesis consists of a list of notations, an introduction, three sections, conclusions and a list of used literature, as well as available appendices.

The explanatory note to the final thesis contains 60 pages, 40 illustrations. The list of used sources contains 25 names and is placed on 3 pages. The total volume of the work is 81 pages.

In the first section, an analysis of the issues was carried out, similar services were considered, a concept document was created and a set of development tools was defined.

In the second chapter, the system architecture was designed and the software code was written.

In the third section, the user interface of the extension and interaction with it were described in detail. The process of testing and elimination of shortcomings has been carried out.

The conclusions section contains general summaries of the service development for simultaneous viewing by remote users.

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						5
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API - Application Programming Interface

TCP - Transmission Control Protocol

JSON - JavaScript Object Notation

БД - база даних

РСУБД - реляційна система управління базами даних

СУБД - система управління базами даних

ID - identity document,

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ.....	6
ВСТУП.....	8
РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ.....	9
1.1 Постановка задачі.....	10
1.2 Аналіз існуючого програмного забезпечення за тематикою кваліфікаційної роботи.....	11
1.3 Концепт-документ (Concept Document).....	14
1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення.....	14
Висновки до першого розділу.....	17
РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	18
2.1 Проектування архітектури системи.....	18
2.2 Розробка бази даних.....	19
2.3 Серверна частина.....	21
2.4 Клієнтська частина.....	25
Висновки до другого розділу.....	37
РОЗДІЛ 3. ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ.....	38
3.1 Опис роботи з програмним додатком (Опис інтерфейсу).....	38
3.2 Тестування роботи програмного забезпечення.....	49
Висновки до третього розділу.....	55
ВИСНОВКИ.....	55
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	58
ДОДАТКИ.....	60
Додаток А.....	61

ВСТУП

Програми для спільного перегляду відеоконтенту є актуальними як у розважальних цілях, так і у роботі.

У такі непрості часи, коли пандемія та війна перешкоджають бути поруч з близькими, здатні розділити не тільки друзів, а й сім'ї, єдиний вихід провести разом час - онлайн-спілкування. Серед онлайн-розваг досить популярним є перегляд фільмів, а сервіси для сумісного перегляду роблять процес легким та приємним.[\[1\]](#)

Вони дозволяють групі друзів або сім'ї разом переглядати фільми, телешоу або інші відео, навіть якщо кожен з них має свій власний пристрій та знаходиться у різних місцях. Це зручний спосіб провести гарно час у колі близьких людей, коли немає можливості зустрітись особисто.

Такі програми можуть використовуватися для віддаленого спільного перегляду відеопрезентацій, групового аналізу певних подій або спільного навчання. Вони допомагають покращити комунікацію та спільну роботу команди, навіть якщо учасники розташовані у різних місцях. У бізнес-середовищі вони стають корисним інструментом для співпраці, підвищення продуктивності та вирішення завдань, що вимагають колективних зусиль.

Загалом, програми для спільного перегляду забезпечують зручний та ефективний спосіб пізнання певної інформації у реальному часі, що може бути критично важливим як у роботі, так і в особистому житті. Вони дозволяють групам людей одночасно насолоджуватися відеоконтентом, обговорювати його та взаємодіяти навіть на відстані. Ці програми сприяють покращенню взаємодії та співпраці між учасниками, що є важливим для ефективної роботи в команді або для покращення якості проведення часу з друзями чи сім'єю.

Існує безліч різних програм для спільного перегляду, серед яких варто виділити такі популярні платформи, як Netflix Party, Kast, TwoSeven та інші. Кожна з цих програм має свої унікальні функції та можливості, що відповідають конкретним потребам та перевагам користувачів. Вибір певної

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

програми залежить від особистих вподобань та специфічних вимог користувачів.

Актуальність теми: зростання популярності стрімінгових платформ, таких як Netflix, YouTube, Hulu тощо, створює великий попит на зручні інструменти для спільного перегляду з друзями чи родиною на відстані.

Мета: розробка зручного сервісу для одночасного перегляду відеоконтенту віддаленими користувачами. Сервіс доступний через веб-браузери у вигляді розширення, що робить його легкою та швидкою альтернативою для групового перегляду.

Об'єкт дослідження: сервіс для одночасного перегляду відеоконтенту віддаленими користувачами

Предмет дослідження: розробка сервісу для одночасного перегляду відеоконтенту віддаленими користувачами.

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

1.1 Аналіз задачі, засобів та методів її вирішення

Основною метою розробки проекту є створення сервісу для одночасного перегляду відеоконтенту віддаленими користувачами. Це має бути зручний та інтуїтивно зрозумілий сервіс, що дозволить одночасно переглядати відеоконтент разом з іншими користувачами, незалежно від їх місцезнаходження. Необхідно забезпечити можливість синхронізації відеопотоків та взаємодії у режимі реального часу, щоб користувачі могли керувати відеопотоком, обговорювати відеоконтент та ділитися враженнями під час перегляду.

Основні завдання, які потрібно вирішити у процесі розробки сервісу, включають:

1. Розробка зручного інтерфейсу, який включає:
 - a) Створення користувача
 - b) Створення кімнати перегляду
 - c) Перегляд доступних кімнат та пошук
 - d) Налаштування
 - e) Вікно взаємодії з відеоконтентом, що містить:
 - i. Чат
 - ii. Список користувачів
 - iii. Керування кімнатою
 - iv. Керування відеопотоком
2. Реалізація механізму синхронізації відеопотоків, щоб усі учасники могли одночасно спостерігати вміст.
3. Інтеграція засобів взаємодії, таких як:
 - a) Чат
 - b) Емоційні реакції
 - c) Повідомлення

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

4. Забезпечення підтримки різних ресурсів відеоконтенту.
5. Розробка серверної частини, що буде опрацьовувати запити користувачів та синхронізувати усі дії між користувачами.

1.2 Аналіз існуючого програмного забезпечення за тематикою кваліфікаційної роботи.

Потреба в сервісах для сумісного перегляду контенту призводить до появи багатьох аналогів, кожен з яких пропонує свої унікальні функції та можливості. Програми для спільного перегляду стали не просто засобом забезпечення спільного часу з родиною та друзями, але й потужним інструментом для комунікації та співпраці в робочих групах. Вони допомагають об'єднувати людей, незалежно від їх географічного розташування, для спільного перегляду відеопрезентацій, аналізу подій чи просто приємного відпочинку.

Важливо провести детальний аналіз цих програм для визначення їх переваг та недоліків. Розглянемо деякі з них:

Netflix Party - це безкоштовне розширення для браузера Google Chrome, яке дозволяє переглядати фільми та телешоу разом із друзями чи родиною віддалено. Воно дозволяє синхронізувати відтворення відео та вступати в чат в режимі реального часу, що робить перегляд більш захоплюючим та веселим. Щоб використовувати Netflix Party, ви і ваші друзі повинні мати активні облікові записи Netflix. Ви можете створити кімнату та запросити друзів за допомогою посилання, яке ви можете надіслати їм електронною поштою або через соціальні мережі. Як тільки всі учасники приєднуються до кімнати, ви можете вибрати фільм або телешоу, яке ви хочете переглянути, та розпочати відтворення. У той же час всі учасники будуть бачити те саме відео і можуть спілкуватися в чаті.[\[2\]](#)

Переваги: зручна взаємодія з розширенням, присутній чат, не має нічого зайвого.

Недоліки: необхідний акаунт Netflix, працює лише з контентом компанії на офіційному сайті.

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		



Рис. 1.1 - Netflix Party

Kast - це онлайн платформа для спільного перегляду відео, включаючи фільми, телешоу та ігри. Також сервіс надає можливість транслювати екран, створювати відеочати та багато іншого. Присутня безкоштовна підписка з обмеженим контентом, середньою роздільною здатністю та інше.[\[3\]](#)

Переваги: зручний інтерфейс, взаємодія з різним контентом, мультиплатформа.

Недоліки: нестабільне підключення, обмеження в браузерній версії.



Рис. 1.2 - Kast

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		12

Watch2gether — це платформа для синхронного потокового відео та аудіо. Вона дозволяє дивитися відео з друзями та родиною, незалежно від того, де вони знаходяться. Сервіс дає можливість створювати списки відтворення та ділитись ними з друзями. Загалом Watch2gether — це зручна платформа для синхронізованого потокового відео та аудіо.[4]

Переваги: доступна безліч сервісів для взаємодії, є можливість транслювати екран, є списки відтворення.

Недоліки: якість відео, реклама, проблеми з синхронізацією.

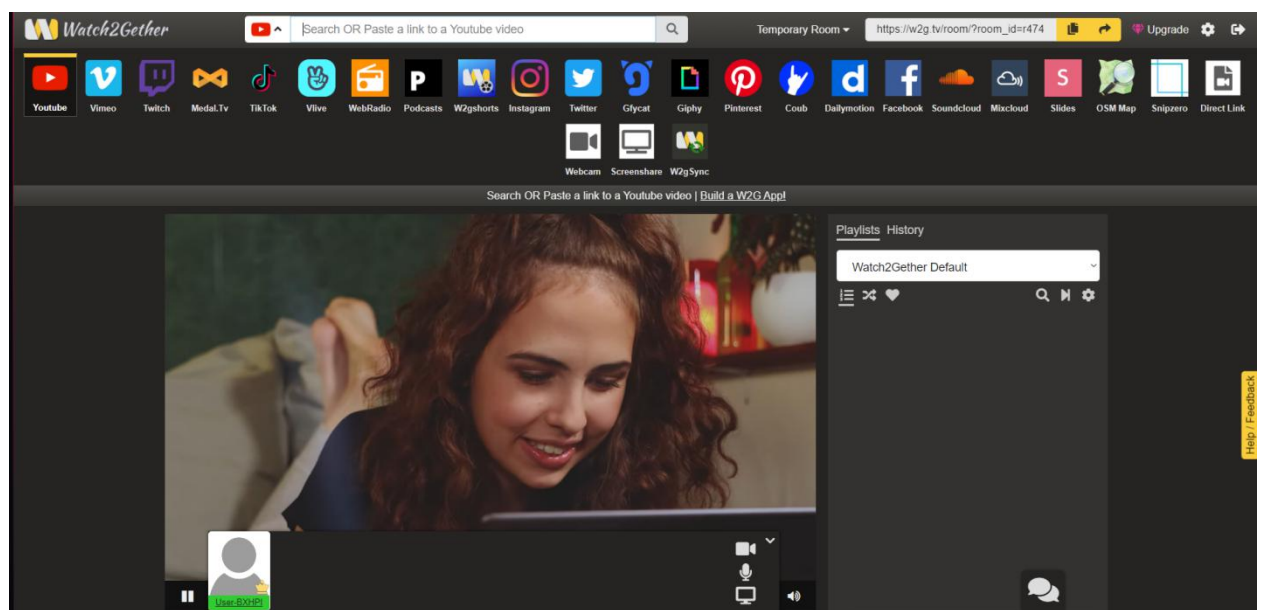


Рис. 1.3 - Watch2gether

На основі проведеного аналізу, стає очевидною важливість розробки власного сервісу для одночасного перегляду відеоконтенту віддаленими користувачами. Саме тому потрібно створити продукт, який буде поєднувати найкращі аспекти існуючих рішень, уникати їх недоліків та надавати користувачам зручний, безперебійний та високоякісний досвід спільного перегляду відеоконтенту в режимі реального часу.

Було виділено основні моменти, які потрібно перейняти чи реалізувати:

1. Приємний візуальний стиль
2. Наявність чату
3. Гарна синхронізація
4. Підтримка різноманітних джерел

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						13
Змн.	Арк.	№ докум.	Підпис	Дата		

5. Швидка реєстрація

Візуальне нагромадження, неприємний інтерфейс, довга реєстрація можуть одразу відштовхнути потенційного користувача, а погана синхронізація може призвести до втрати вже наявних користувачів. Тому слід уникати цих аспектів.

1.3 Концепт-документ (Concept Document)

Назва:

“4eyes”

Короткий опис:

Розширення для браузерів, що дозволяє переглядати відеоконтент з певною групою людей на відстані.

Аудиторія:

Обмежень немає

Основні особливості розширення:

Можливість синхронно використовувати улюблені сервіси для перегляду контенту.

Передумови створення:

Кожен, хоча б раз, хотів переглянути певний фільм з друзями, які далеко від тебе. Тут постає питання: А як запустити відео одночасно? Які сервіси підійдуть? Де буде наявний необхідний контент? На ці та інші запитання існує відповідь - “4eyes”.

Платформа:

Розширення розроблено під браузери на базі Chromium. Доступність платформи визначається наявністю браузера на базі Chromium.

Доступні: Windows, MacOS, Linux, Android.

1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для розробки клієнтської частини сервісу, яка є відповідальною за взаємодію з користувачем, вибрано такі інструментальні засоби, як JavaScript,

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

HTML та CSS. Цей набір технологій є стандартним для розробки веб-додатків і забезпечує широкі можливості для створення інтерактивного та привабливого інтерфейсу користувача.[\[5\]](#)

Щодо серверної частини сервісу, вибір пав на мову програмування C#. Ця мова є потужною та масштабованою. C# має широку підтримку та багато інструментів, що дозволило ефективно реалізувати бізнес-логіку, забезпечити безпеку та обробку запитів від клієнтів. Також C# є компільованою мовою, що дозволяє досягти високої продуктивності виконання програм. Вона має ефективний збірник сміття та оптимізовану роботу з пам'яттю, що важливо для забезпечення швидкої та надійної роботи нашого сервера. Враховуючи всі ці фактори, C# є ідеальним вибором для реалізації серверної частини сервісу, надаючи потужність, гнучкість та широкі можливості для розвитку та підтримки продукту.[\[6\]](#)

“Спілкування” клієнтів з сервером реалізовано через WebSockets. Основні причини використання WebSockets включають:

1. Двосторонній зв'язок - WebSockets дозволяє встановлювати постійне двостороннє з'єднання між клієнтом та сервером. Це означає, що обидва боки можуть ініціювати передачу даних без очікування запиту від іншої сторони. Це дозволяє нам надсилати оновлення та повідомлення миттєво, що особливо важливо для спільного перегляду відеоконтенту, де синхронізація відтворення має вирішальне значення.
2. WebSockets використовують мінімальну кількість протоколів і накладних витрат, що дозволяє досягти високої швидкості передачі даних та зниження затримок. Це особливо важливо для нашого сервісу, де реагування в реальному часі є ключовим фактором. WebSockets базується на стандартному протоколі TCP, що робить його сумісним з різними серверами та засобами розробки.
3. Підтримка браузерами - WebSockets підтримується більшістю сучасних веб-браузерів, що дозволяє забезпечити сумісність та доступність нашого сервісу для широкого кола користувачів.[\[7\]](#)

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

Такий набір забезпечує мінімальне навантаження, миттєву взаємодію між користувачами, а розробка та майбутні оновлення не будуть викликати певних труднощів.

Основною вимогою до пристроїв є лише підтримка браузера, створеного на основі Chromium. Найпопулярніші з них: Google Chrome, Opera, Chromium, Brave, Microsoft Edge, Epic Browser тощо.[\[8\]](#)

В якості СУБД використовуватимемо PostgreSQL. PostgreSQL являється однією з найпродвинутіших БД та має ряд таких переваг:

Модель даних

PostgreSQL не просто реляційна, а об'єктно-реляційна СУБД. Це дає йому деякі переваги над іншими базами даних SQL з відкритим вихідним кодом, такими як MySQL, MariaDB і Firebird.

Фундаментальна характеристика об'єктно-реляційної бази даних — це підтримка об'єктів користувача та їх поведінки, включаючи типи даних, функції, операції, домени та індекси. Це робить Постгрес неймовірно гнучким та надійним. Серед іншого, він вміє створювати, зберігати та отримувати складні структури даних. Можна створювати вкладені та складові конструкції, які не підтримуються стандартними РСУБД.

Структури та типи даних

Існує великий список типів даних, які підтримує постгрес. Крім числових, з плаваючою точкою, текстових, булевих та інших очікуваних типів даних (а також безлічі їх варіацій), PostgreSQL може похвалитися підтримкою uuid, грошового, перерахованого, геометричного, бінарного типів, мережесов'язаних адрес, бітових рядків, текстового пошуку, xml, масивів, композитних типів та діапазонів, а також деяких внутрішніх типів для ідентифікації об'єктів та розташування логів. Заради справедливості варто сказати, що MySQL, MariaDB і Firebird теж мають деякі з цих типів даних, але тільки Постгрес підтримує їх усі.

Цілісність даних

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

Постгрес прагне відповідати стандарту ANSI-SQL:2008, відповідає вимогам ACID (атомарність, узгодженість, ізолюваність та надійність) та відомий своєю посиляльною та транзакційною цілісністю. Первинні ключі, що обмежують та каскадні зовнішні ключі, унікальні обмеження, обмеження NOT NULL, перевірочні обмеження та інші функції забезпечення цілісності даних дають впевненість, що лише коректні дані будуть збережені.[\[9\]](#)

Для взаємодії з БД можна використати Entity Framework. Він представляє спеціальну об'єктно-орієнтовану технологію на базі фреймворку .NET для роботи з даними. Якщо традиційні засоби ADO.NET дозволяють створювати підключення, команди та інші об'єкти для взаємодії з базами даних, то Entity Framework є вищим рівнем абстракції, який дозволяє абстрагуватися від самої бази даних і працювати з даними незалежно від типу сховища. Якщо фізично ми оперуємо таблицями, індексами, первинними та зовнішніми ключами, але на концептуальному рівні, який нам пропонує Entity Framework, ми вже працюємо з об'єктами.[\[10\]](#)

Підсумовуючи вище сказане, цей фреймворк дозволяє створити взаємозв'язок між БД та класами програми. В його можливості також входить і автоматичне створення таблиць на основі класів.

Висновки до першого розділу

В першому розділі було сформульовано основну мету розробки проекту - розробку сервісу для одночасного перегляду відеоконтенту віддаленими користувачами.

Було проведено аналіз існуючого програмного забезпечення за тематикою кваліфікаційної роботи. Також був створений концепт-документ, який допоміг визначити потреби ринку та конкретизувати вимоги до сервісу. Було обґрунтовано вибір інструментальних засобів та вимоги до апаратного забезпечення. Спираючись на проведений аналіз, було обрано список інструментальних засобів розробки та вимоги до апаратного забезпечення.

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Проектування архітектури системи

У рамках даного даної кваліфікаційної роботи була розроблена архітектура системи, яка базується на клієнт-серверній моделі.

У цій моделі сервер є центральним елементом системи. Він обробляє запити від клієнтів і забезпечує їм необхідні ресурси або послуги. Сервер може мати різні функції і здійснювати обробку даних, зберігання, обчислення, комунікацію з іншими серверами тощо. Лише сервер має доступ до бази даних і тільки він має можливість додавати, змінювати чи видаляти дані.

Клієнт може бути будь-якою програмою або пристроєм, що взаємодіє з сервером для отримання певних послуг або ресурсів. У нашому випадку клієнтом є веб-браузер користувача, що надсилає запити на сервер

Клієнт-серверна модель дозволяє розподіляти функціональність та навантаження між клієнтом і сервером, сприяючи масштабованості та ефективності системи. Клієнти можуть бути розташовані на різних пристроях або в мережі, а сервери можуть бути централізовані або розподілені. Ця модель дозволяє розробляти складні системи з розподіленою обробкою даних та взаємодією між користувачами та серверами.[\[11\]](#)

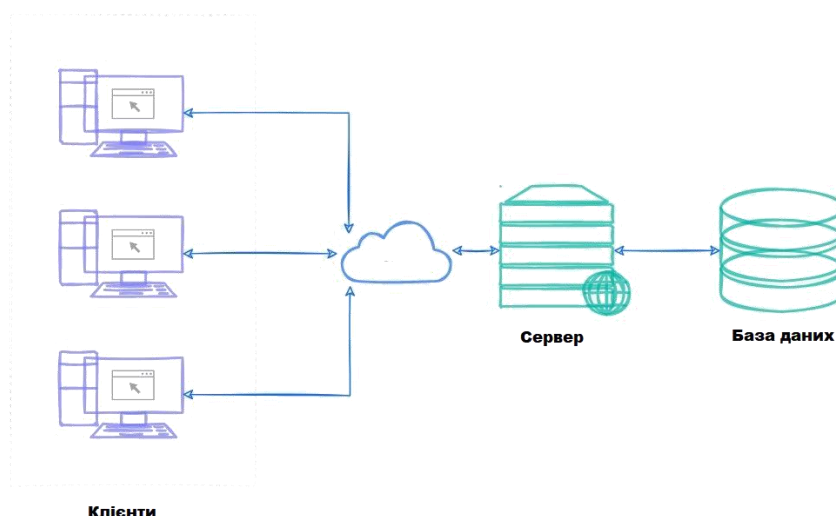


Рис. 2.1 - Схема взаємодії системи

2.2 Розробка бази даних

Для реалізації сервісу, де наявні користувачі, обов'язково повинна бути база даних. У проекті буде достатньо трьох таблиць, а саме:

1. User - для збереження користувачів
2. Room - для збереження інформації кімнат
3. UserRoom - для реалізації з'єднання багато до багатьох

Деякі дані, такі як активні підключення, немає сенсу зберігати в БД, тому частина інформації зберігатиметься локально. Маємо наступний вигляд БД:

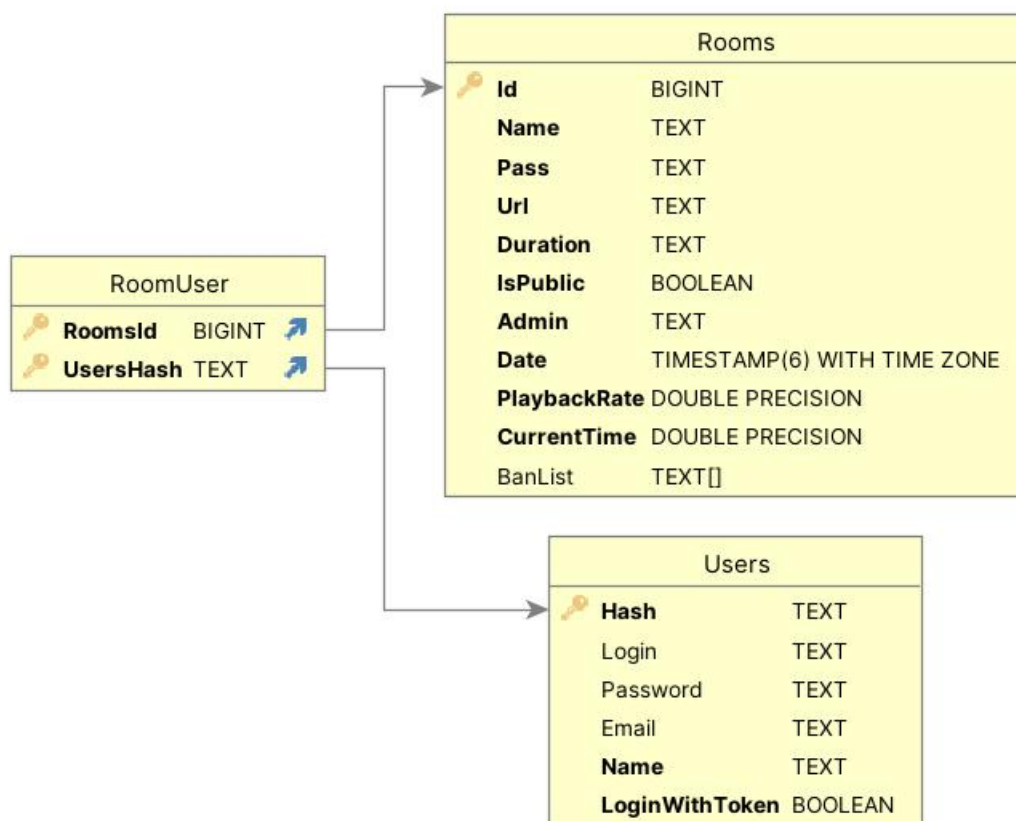


Рис. 2.2 - Схема БД

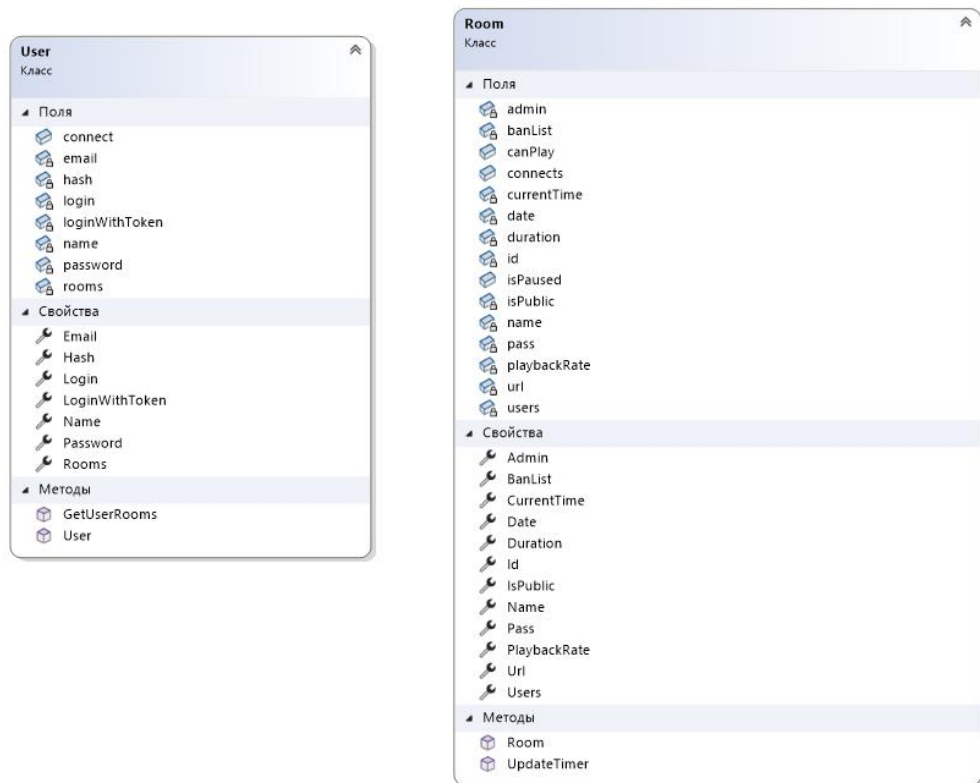


Рис. 2.3 - Діаграма класів

Було створено клас для взаємодії з БД:

```

using Microsoft.EntityFrameworkCore;

namespace SocketServer
{
    Ссылка 12
    public class DataBase : DbContext
    {
        Ссылка 4
        private static DataBase db;
        Ссылка 4
        private DbSet<User> Users { get; set; }
        Ссылка 4
        private DbSet<Room> Rooms { get; set; }

        Ссылка 1
        private DataBase()
        {
            Database.EnsureCreated();
        }
        Ссылка 0
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseNpgsql("Host=localhost;Port=5432;Database=4eyes;Username=postgres;Password=123");
        }

        Ссылка 4
        public static DataBase ConnectToDB()
        {
            if (db == null)
            {
                db = new DataBase();
                var b12 = db.Rooms.Include(c => c.Users).ToList();
                return db;
            }
            else
            {
                return db;
            }
        }
    }
}

```

За допомогою патерна «Одинак» створюється підключення до БД у методі ConnectToDB(). В методі OnConfiguring(..) прописуються налаштування підключення. Також клас містить методи для взаємодії з даними:

```

Ссылка: 3
public void AddUser(User user)
{
    User newUser = GetUser(user.Hash);
    if (newUser == null)
    {
        db.Users.Add(user);
        db.SaveChanges();
    }
    db.SaveChanges();
}

Ссылка: 5
public User GetUser(string hash)
{
    return db.Users.FirstOrDefault(x => x.Hash == hash);
}

Ссылка: 1
public User GetUserbyLogin(string login)
{
    return db.Users.FirstOrDefault(x => x.Email == login);
}

Ссылка: 0
public List<User> GetAllUsers()
{
    return db.Users.ToList();
}

Ссылка: 1
public Room AddRoom(string Name, string Pass, string Url, string Admin, bool isPublic)
{
    long id;
    Random rnd = new();
    do
    {
        id = rnd.Next(100000, 999999);
    } while (db.GetRoom(id) != null);
    Room room = new Room(id, Name, Pass, Url, Admin, isPublic);
    db.Rooms.Add(room);
    db.SaveChanges();
    return room;
}

Ссылка: 5
public Room GetRoom(long id)
{
    return db.Rooms.FirstOrDefault(x => x.Id == id);
}

```

2.3 Серверна частина

Як і протокол HTTP, WebSocket має звичайне та захищене з'єднання. Підключення WSS (WebSocket Secure) та WS (WebSocket) є протоколами для забезпечення двостороннього зв'язку між клієнтом та сервером через веб-

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

сокети. WSS використовує шифрування SSL/TLS для захищеної передачі даних, тоді як WS працює через незахищений з'єднання. Обидва протоколи забезпечують постійне з'єднання між клієнтом та сервером, що дозволяє взаємодіяти в реальному часі та передавати дані у двох напрямках. Щоб використовувати WebSocket на захищених сайтах, він теж повинен мати захист, тобто використовувати WSS підключення.[\[12\]](#)

Для можливості безпечного підключення необхідно отримати SSL-сертифікат. Це можна зробити через сертифікаційні органи, які видають платні сертифікати, або використовуючи безкоштовні сервіси, наприклад, Let's Encrypt. В сертифікаті вказується адреса сервера, а саме доменне ім'я, яке теж варто зареєструвати у відповідних сервісах. Усі ці кроки потрібно проробити для забезпечення безпечного підключення та передачі даних між клієнтом та сервером за допомогою шифрування SSL/TLS.[\[13\]](#)

Реалізований виконавчий клас Program для сервера:

```
namespace SocketServer
{
    Ссылка 0
    public class Program
    {
        Ссылка 0
        public static void Main(string[] args)
        {
            var wssv = new WebSocketServer(8080);
            // var wssv = new WebSocketServer(8080, true);
            string fileds = Directory.GetCurrentDirectory() + "/sert.pfx";
            Console.WriteLine(fileds);
            //wssv.SslConfiguration.ServerCertificate = new X509Certificate2(fileds, "Aboba34");
            //wssv.Log.Level = LogLevel.Trace;
            //wssv.SslConfiguration.EnabledSslProtocols = SslProtocols.Tls12;
            //ServicePointManager.ServerCertificateValidationCallback += (object sender, X509Certificate cer, X509Chain chain,
            // {
            //     return true;
            // };
            wssv.AddWebSocketService<Connect>("/connect");
            wssv.AddWebSocketService<Chat>("/chat");
            wssv.AddWebSocketService<Login>("/login");
            wssv.Start();
            if (wssv.IsListening)
            {
                Console.WriteLine("Listening on port {1}:{0}, and providing WebSocket services:", wssv.Port, wssv.Address);
                foreach (var path in wssv.WebSocketServices.Paths)
                {
                    Console.WriteLine("- {0}", path);
                }
            }
            Console.WriteLine("\nPress Enter key to stop the server...");
            Console.ReadLine();
            wssv.Stop();
        }
    }
}
```

У цьому класі створюється екземпляр WebSocketServer, використовуючи стандартний порт для TCP з'єднання. Після цього прописано налаштування безпечного підключення WSS, але під час розробки ця частина залишатиметься закоментованою. Після налаштувань безпеки додаються потрібні сервіси, які

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		22

створюються окремим класом та кожному прописується шлях, за яким можна створити підключення. Після цього сервер запускається і очікує підключення.[\[14\]](#)

Сервіси для WebSocket мають основні методи для користування, а саме:

- OnOpen() - при відкритті з'єднання
- OnClose() - при закритті з'єднання
- OnMessage() - для обробки вхідних повідомлень
- OnError() - при наявних помилках

Розглянемо клас Connect:

```
Ссылка: 0
protected override void OnOpen()
{
    var hash = QueryString["hash"];
    var user = db.GetUser(hash);
    if(user == null)
    {
        user = new User(hash, "Unknown User");
        db.AddUser(user);
    }

    if(user.connect == null)
    {
        user.connect = this;
        this.user = user;
    }
    else
    {
        dynamic data = new JObject();
        data.status = "Re-entry";
        this.user = user;
        Sessions.SendTo(GetJSON(Status.Connect, JsonConvert.SerializeObject(data)), this.ID);
        return;
    }
    SendUserInfo();
}
```

При створенні підключення на сервер передається ідентифікатор користувача. Обробляються можливі помилки та, якщо все добре, повертається інформація користувача.

Також варто зазначити, що користувач може авторизуватися лише з одного пристрою. Це зроблено для того, щоб не виникало проблем з підключенням до кімнати.

```
protected override void OnMessage(MessageEventArgs e)
{
    dynamic M = JsonConvert.DeserializeObject(e.Data);
    int code = int.Parse($"{M.code}");
    string msg = $"{M.msg}";
    switch (code)
    {
        case -1:
            ...
        case 0://отримати список кімнат
            ...
        case 1://створити кімнату
            ...
        case 2://увійти до кімнати
            ...
        case 3:
            ...
        case 4:
            ...
        case 5:
            ...
        /* ...
    }
}
```

Усі комунікації відбуваються в форматі JSON. За допомогою JsonConvert повідомлення перетворюється в об'єкт для зручності використання. Кожне повідомлення має свій код та тіло, яке теж може бути у вигляді JSON.

Ссылка 0

```
protected override void OnClose(CloseEventArgs e)
{
    if(user != null)
    {
        if(user.connect.ID == ID)
            user.connect = null;
    }
}
```

OnClose(..) видаляє закриті підключення.

Щоб зручно надсилати відповідь був написаний метод:

Ссылка 12

```
private string GetJSON(Status code, string data, string[] args = null)
{
    int coded = (int)code;
    var o = JObject.Parse($"{{'code':{coded}}}"");
    o["data"] = JToken.FromObject(data);
    if (args != null)
    {
        if(args.Length % 2 == 0)
        {
            for(int i = 0; i < args.Length - 1; i++)
            {
                o[args[i]] = JToken.FromObject(args[++i]);
            }
        }
    }
    return o.ToString();
}
```

Він формує шаблон повідомлень та має можливість вказати додаткові аргументи.

Інші класи написані подібним чином та не потребують окремого розгляду на даному етапі.

2.4 Клієнтська частина

Клієнтська сторона системи базується на веб-технологіях, таких як HTML, CSS та JavaScript. Веб-браузер виконує роль клієнта, який взаємодіє з користувачем та надає зручний інтерфейс для спілкування з системою. Використання WebSocket дозволяє встановлювати двостороннє з'єднання між клієнтом і сервером, забезпечуючи миттєву передачу даних. Усі технології є стандартними, що допомагає уникнути більшості проблем з сумісністю. При розробці клієнтської частини сервісу можна виокремити такі частини:

1. Реєстрація
2. Створення кімнат
3. Інтерфейс для перегляду

Будь-яке розширення має певну структуру та ряд обов'язкових файлів. Найголовніший - це manifest.json. У цьому файлі вказується основна інформація про розширення, необхідні дозволи та задіяні ресурси. Він має такий вигляд:

```
"name": "4eyes dev",
"manifest_version": 3,
"version": "0.1.0",
"permissions": ["scripting", "activeTab", "storage", "tabs"],
"icons": {
  "16": "icon.png",
  "32": "icon.png",
  "48": "icon.png",
  "128": "icon.png"
},
"host_permissions": [
  "*/**/*"
],
"action": {
  "default_popup": "popup.html"
},
```

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

“name” - назва розширення, за яким можна його відшукати в магазині розширень.

"manifest_version" - цей параметр вказує на версію маніфесту (manifest), який визначає налаштування та характеристики розширення. Він є дуже важливим, оскільки версія маніфесту може визначати підтримувані функції та можливості розширення, а також впливати на сумісність з різними версіями браузера. У нашому випадку обрана найновіша.

"version": - версія програми. Допомогає орієнтуватись в оновленнях.

"permissions" - вказані необхідні дозволи для розширення. А саме:

1. "scripting" - для можливості додавати скрипти на сторінку,
2. "activeTab" - для можливості отримувати активну вкладинку та взаємодіяти з нею,
3. "storage" - для можливості зберігати та діставати дані зі сховища,
4. "tabs" - для взаємодії з вкладинками браузера.

"icons" - вказуються усі іконки розширення різних розмірів.

"host_permissions" - використовується, щоб вказати на яких сторінках розширення матиме доступ.

"action" - вказує з чого починає свою роботу розширення. Існують сценарії, коли інтерфейс не потрібен, тоді там вказується файл зі скриптом.

Для початку цього буде достатньо, в майбутньому його можна розширити, вказавши потрібні дані.

Перш за все, опираючись на зручність користувачів, було розроблено можливість зміни мови інтерфейсу. Це важлива функціональність, яка забезпечує доступність та зручність використання сервісу для користувачів з різних країн та мовних груп. Було розроблено клас, який керує масивами текстових даних.

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

```

var userLang = navigator.language || navigator.userLanguage;
var lang;

if(userLang.match( regexp: "ua")) lang = SetLanguage( lang: "ua");
else lang = SetLanguage( lang: "en");

const availableLanguages = [
    { code: 'ua', name: 'Українська' },
    { code: 'en', name: 'English' },
];

5+ usages sarakamotaka *
function SetLanguage(lang){
    switch (lang) {
        case "ua":...
        case "en":...
    }
}

```

Він реалізований так, щоб можна було додавати нові мови без зміни самого коду. Відповідно до мови інтерфейсу користувача, обирається значення за допомогою switch. Якщо мови користувача немає серед доступних, то встановлюється англійська по замовчуванню. Тепер можна використовувати змінну lang, обираючи необхідний елемент. Щоб можна було оновлювати текстову частину інтерфейсу, потрібно додати метод, що робитиме це:

```

function UpdateInterfaceText(){
    document.getElementById( elementId: 'infoToken').title = lang['HashW'];
    let el = document.getElementsByClassName( classNames: 'lang')
    for (let i = 0; i < el.length; i++) {
        el[i].innerText = lang[el[i].dataset.name];
    }
    el = document.getElementsByClassName( classNames: 'beforeValue')
    for (let i = 0; i < el.length; i++) {
        el[i].dataset.value = lang[el[i].dataset.name];
    }
}

```

Усі елементи з текстом будуть помічені спеціальним параметром, що дозволить оновити одразу весь текст. Також потрібно врахувати текст, який

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

знаходиться не в середині тегу, а, наприклад, в псевдокласі before. Для нього створений окремий клас, який буде брати дані зі спеціального параметра.

Тепер можна перейти до побудови інтерфейсів. Кожного користувача має зустрічати вікно реєстрації/входу, тому з нього і варто почати:

Language: Українська

Згенерувати новий нікнейм

Спочатку придумайте собі нік

perplexed Beetle

Зареєструватися

Можливо...

Вже є аккаунт? Увійти

Рис. 2.4 - Вікно реєстрації

Назад

Пошта: 4eyes@mail.com

Пароль: MyStrongPass

Увійти

Бажаєте увійти з допомогою хеша? f1111ff1-1111-1f111-f111-1f1f1 Відновити

Рис. 2.5 - Вікно входу

Також варто забезпечити безпеку інформації, такий як паролі, від зовнішніх спостерігачів, тому такі поля мають тип password. Щоб переглянути введене додані спеціальні кнопки:

Пошта: 4eyes@mail.com

Пароль:

Увійти

Рис. 2.6 - Введення пароля

Пошта: 4eyes@mail.com

Пароль: перевірка

Увійти

Рис. 2.7 - Перегляд введення пароля

Усі підключення будуть реалізовані через WebSocket. Щоб не встановлювати непотрібні підключення, з'єднання з сервером відбуватиметься по необхідності.

Підключення для реєстрації відбувається через функцію Login():

```
function Login(){
    websocket = new WebSocket( url: url + 'login');
    websocket.onmessage = function (event : MessageEvent ) {
        const obj = JSON.parse(event.data);
        console.log('code:' + obj.code);
        switch (obj.code){
            case 0:{
                storage.set({
                    eyesId: obj.data.hash,
                    eyesName : obj.data.name
                });
                Connect();
                ActiveTab(PageTab.main);
                break;
            }
            case -1:{
                GenerateErrorInfo(lang['Error'], info: '');
                break;
            }
        }
    };
}
```

Вона може отримати підтвердження або помилку.

При успішній авторизації відбувається підключення до основного блоку розширення через функцію Connect():

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

```

function Connect(){
    storage.get("eyesId",function(items){
        let hash = items.eyesId;
        if(!items.eyesId) hash = 'new';
        websocket = new WebSocket( url: url + 'connect?hash=' + hash);
        Loading(document.getElementById( elementId: 'rooms'));
        websocket.onmessage = function (event : MessageEvent ) {
            const obj = JSON.parse(event.data);
            console.log('code:' + obj.code);
            switch (obj.code){
                case -1:...
                case 0:...
                case 1:...
                case 2:...
                case 3:...
                case 4:...
                case 5:...
                case 6:...
            }
        };
        websocket.onerror = function (event : Event ) {
            GenerateErrorPage(lang['ErrorConnect'], lang['ReConnect'], Reconnect);
        };

        websocket.onclose = function (event : CloseEvent ) {
            if(reconnectCount > 5) return;
            reconnectCount++;
            Reconnect();
            setTimeout( handler: () => { }, timeout: 5000);
        };
    });
}

```

При підключенні передається ідентифікатор користувача. Була додана обробка помилки підключення. На сторінці помилки можна спробувати підключитись ще раз, якщо автоматичне перепідключення не вдалось.

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		



Рис. 2.8 - Анімація заднього фону

Усі підключення та завантаження супроводжуються анімаціями. Щоб заповнити пустоту, додана анімація заднього фону у вигляді ока.[\[15\]](#)

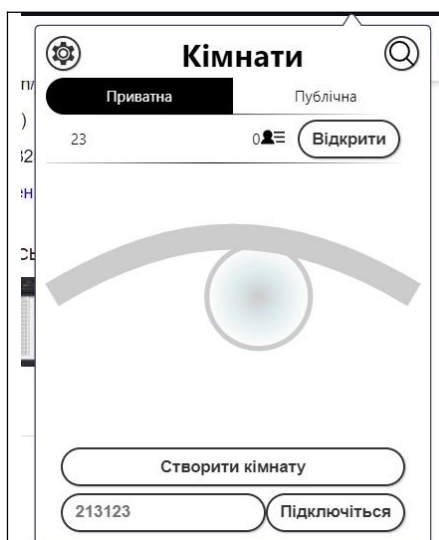


Рис. 2.9 - Анімація заднього фону

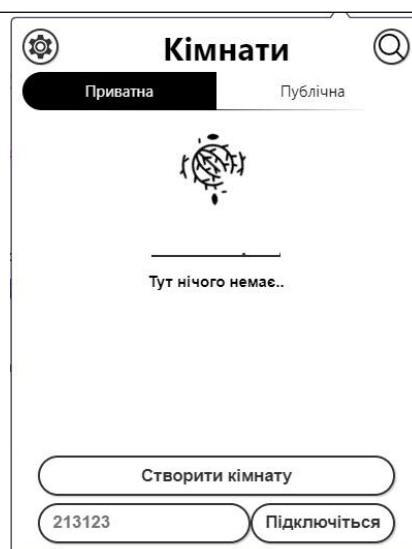


Рис. 2.10 - Анімація при відсутності кімнат

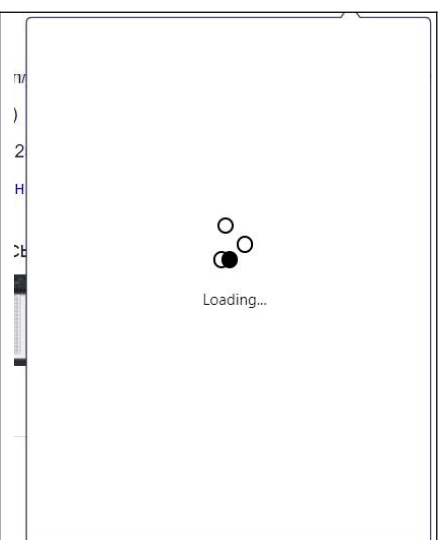


Рис. 2.11 - Анімація завантаження

Були додані такі сторінки:

Рис. 2.12 - Інформація кімнати

Рис. 2.13 - Створення кімнати

Рис. 2.14 - Налаштування

На сторінках багато об'єктів, які змінюють зовнішній вигляд, тому потрібно створено функцію, яка повертатиме їх у початковий стан при перемиканні сторінок:

```
function ActiveTab(tab){
  if ([0,4,5,7].includes(tab)){
    document.getElementById( elementId: 'Header').classList.add('vis');
  }else{
    document.getElementById( elementId: 'Header').classList.remove( tokens: 'vis');
  }
  let children = document.getElementById( elementId: 'main').children;
  for (let i = 0; i < children.length; i++) {
    children[i].classList.remove( tokens: 'active');
  }
  children[tab].classList.add('active');
  document.getElementById( elementId: 'pageName').innerText = lang[children[tab].dataset.name];
  let back = document.getElementById( elementId: 'backButton');
  let find = document.getElementById( elementId: 'FilterButton');
  if(tab === 1){
    back.style.display = 'none';
    find.style.display = 'flex';
  }else {
    find.style.display = 'none';
    back.style.display = 'flex';
  }
  let a = document.getElementById( elementId: 'deleteRoom');
  if (a.innerText !== lang['Delete']){
    a.click();
  }
  document.getElementById( elementId: 'roomId').value = '';
  document.getElementById( elementId: 'SearchAndFilter').classList.remove( tokens: 'max');
}
```

При введенні неправильний даних, користувача повідомляють про це спливаючим вікном:

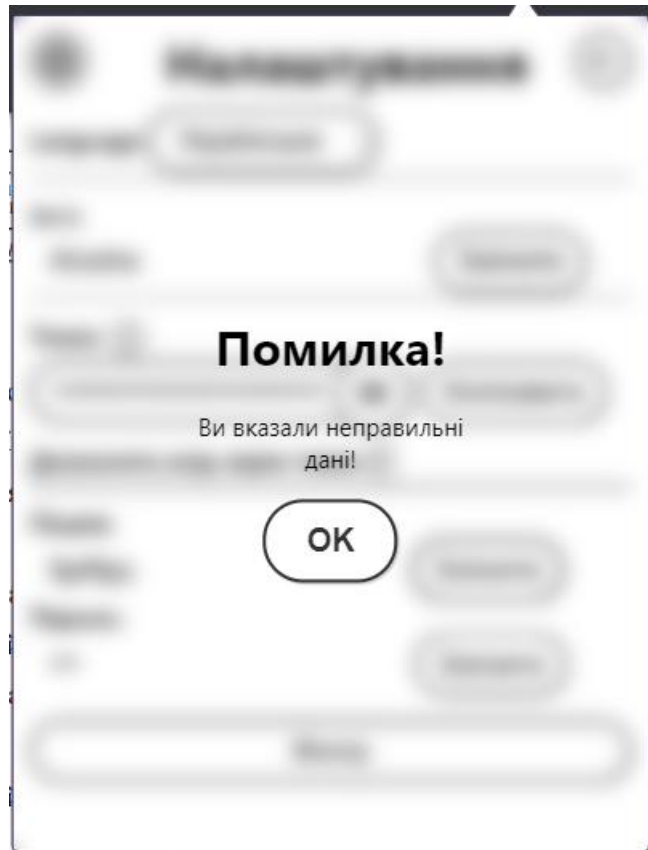


Рис. 2.15 - Повідомлення про помилку введених даних

При підключенні до кімнати відкривається відповідна сторінка браузера (якщо активна сторінка не відповідає посиланню кімнати) та вводиться на сторінку сайту код:

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						33
Змн.	Арк.	№ докум.	Підпис	Дата		

```

function injectScript() {
  let info = document.getElementById( elementId: 'RoomInfo');
  let urlR = info.children[2].children[0].value;
  storage.set({
    eyesRoom:info.children[1].innerText
  },function () {
    chrome.tabs.query({active: true, currentWindow: true}, (tabs) => {
      console.log(urlR)
      const tab = tabs[0];
      if (tab.url === urlR) {
        chrome.scripting.executeScript({
          target: {tabId: tab.id},
          files: ["Player.js", "Vocabulary.js", "pageScript.js"],
        });
      } else {
        chrome.tabs.create({url: urlR}, async function (tab) {
          await chrome.tabs.update(tab.id, {active: true});
          await chrome.scripting.executeScript({
            target: {tabId: tab.id},
            files: ["Player.js", "Vocabulary.js", "pageScript.js"],
          });
        });
      }
    });
  });
}

```

Щоб взаємодіяти з плеєром розроблені функції:

```

function Pause(){
  eyesPlayer.onpause = null;
  eyesPlayer.pause();
  SentPlayerInfo( type: 'status', value: false);
  setTimeout( handler: ()=>{
    eyesPlayer.onpause = PlayerPause;
  },
    timeout: 200)
}
3 usages sarakamotaka *
function Play(){
  eyesPlayer.onplay = null;
  eyesPlayer.play();
  SentPlayerInfo( type: 'status', value: true);
  setTimeout( handler: ()=>{
    eyesPlayer.onplay = PlayerPlay;
  }, timeout: 200)
}

```

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						34
Змн.	Арк.	№ докум.	Підпис	Дата		

```

2 usages sarakamotaka *
function PlayerPlay(){
    if(isSeeking === true) return;
    Pause();
    eyesPlayer.onplay = PlayerWait;
    SentPlayerInfo( type: 'play', value: true, code: 4);
}

2 usages sarakamotaka
function PlayerPause(){
    if(isSeeking === true) return;
    Play();
    SentPlayerInfo( type: 'play', value: false, code: 4);
}

4 usages sarakamotaka *
function ChangeDuration(time){
    eyesPlayer.currentTime = time;
}

```

При спробі увімкнути відео чи натиснути паузу, ця можливість блокується для користувача. Після цього відправляється на сервер запит. Якщо користувачу дозволено зробити цю дію, сервер одночасно усім надсилає команду. Завдяки такому підходу досягається максимальна синхронізація.

```

function DeletePlayerEvent(){
    eyesPlayer.onplay = null;
    eyesPlayer.onpause = null;
    eyesPlayer.onseeked = null;
    eyesPlayer.onseeking = null;
}

2 usages sarakamotaka
function UpdatePlayerEvent() {
    eyesPlayer.onplay = PlayerPlay;
    eyesPlayer.onpause = PlayerPause;
    eyesPlayer.onseeked = Seeked;
    eyesPlayer.onseeking = Seeking;
    eyesPlayer.onratechange = RateChange;
}

```

Як тільки був отриманий відеоплеєр, викликається функція UpdatePlayerEvent(), що підписується на події відеоплеєра. Якщо втрачається зв'язок чи користувач сам виходить, спрацьовує DeletePlayerEvent(), який видаляє усі підписки.

```

function FindPlayer(){
    let videoPlayers = [];
    let player = document.getElementsByTagName(qualifiedName: 'video');
    videoPlayers.push(...player);
    let frames = document.getElementsByTagName(qualifiedName: 'iframe');
    let url = document.location.href;
    for (let i = 0; i < frames.length; i++) {
        try {
            var iframeWindow = frames[i].contentWindow;
            var iframeElement = iframeWindow.document.getElementsByTagName(qualifiedName: 'video');
            videoPlayers.push(...iframeElement);
        } catch {
        }
    }
}

```

Пошук відеоплеєра відбувається не тільки на основній частині сайту, а й в кожному фреймі, бо зазвичай плеєр є окремим елементом і потребує ізолюваного простору.

Також для обробки та відправки інформації про події написана функція:

```

function SendPlayerInfo(type, value, code : number = 4){
    if(websocket){
        let temp = {
            type: type,
            value: value,
            time: eyesPlayer.currentTime
        }
        send(JSON.stringify(temp), code);
    }
}

```

Відправляється лише необхідна інформація, що додатково може вплинути на швидкодію сервісу.

Висновки до другого розділу

У результаті розробки другого розділу було створено повноцінну архітектуру системи, яка відповідає вимогам та потребам ринку.

Була розроблена база даних, яка містить усі необхідні змінні, а також описано основні реалізовані методи.

Було розроблено серверну та клієнтську частину сервісу, а також їх взаємодія.

В результаті, було створено повноцінний сервіс, який може бути використаний для одночасного перегляду відеоконтенту віддаленими користувачами.

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						37
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ

3.1 Опис роботи з програмним додатком (Опис інтерфейсу)

Для встановлення розширення достатньо перетягнути його на вікно браузера та натиснути «встановити». В майбутньому планується додати в магазин розширень, щоб встановлювати через офіційні ресурси.

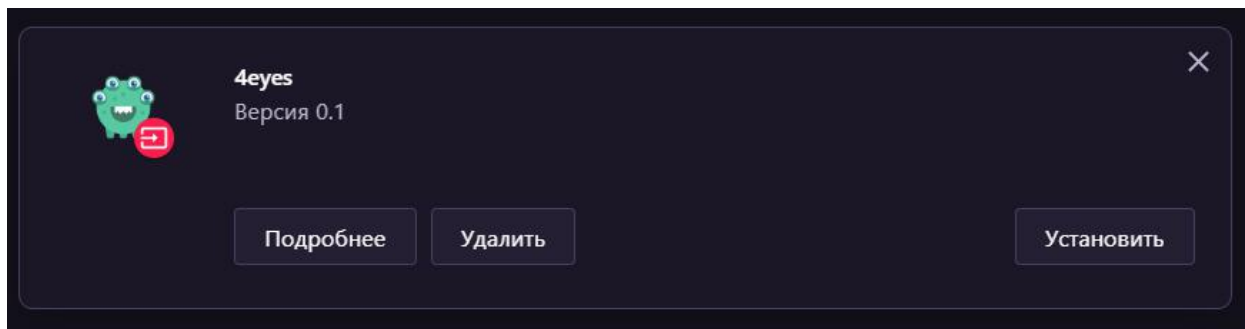


Рис. 3.1 - Встановлення розширення

Тепер його можна запустити на будь-якій сторінці з панелі розширень.

Запустивши вперше, користувача зустрічає вікно реєстрації. На ньому можна одразу змінити мову інтерфейсу, якщо не влаштовує автоматичний вибір. Усе що потрібно для реєстрації - це нікнейм. До того ж вже є прописаний варіант і можна натиснути лише одну кнопку «Зареєструватися». Якщо немає бажання придумувати собі нікнейм, а запропонований не сподобався, то праворуч є кнопка генерації нових імен. Така простота зроблена для того, щоб мати можливість зацікавити користувача. Це можна порівняти з режимом «гостя» в інших програмах. Зареєстрований користувач зможе в майбутньому додати, звичні для багатьох, логін та пароль. При реєстрації, кожному користувачеві присвоюється унікальний ідентифікатор, за допомогою якого можна увійти в існуючий профіль.

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						38
Змн.	Арк.	№ докум.	Підпис	Дата		



Language: Українська

Згенерувати новий нікнейм

Спочатку придумайте собі нік

mysterious garter snake

Зареєструватися

Можливо...

Вже є аккаунт? Увійти

Рис. 3.2 - Вікно реєстрації

Внизу можна побачити кнопку увійти. Натиснувши, користувач потрапляє у вікно авторизації. На ньому є кнопка повернення назад, форма для входу, використовуючи логін та пароль, а також рядок для входу через унікальний токен.



Назад

Пошта:
4eyes@mail.com

Пароль:
MyStrongPass

Увійти

Бажаєте увійти з допомогою хеша?
f1111ff1-1111-1f111-f111-1f1f1' Відновити

Рис. 3.3 - Вікно авторизації

Авторизувавшись, користувача зустрічає головне вікно розширення:

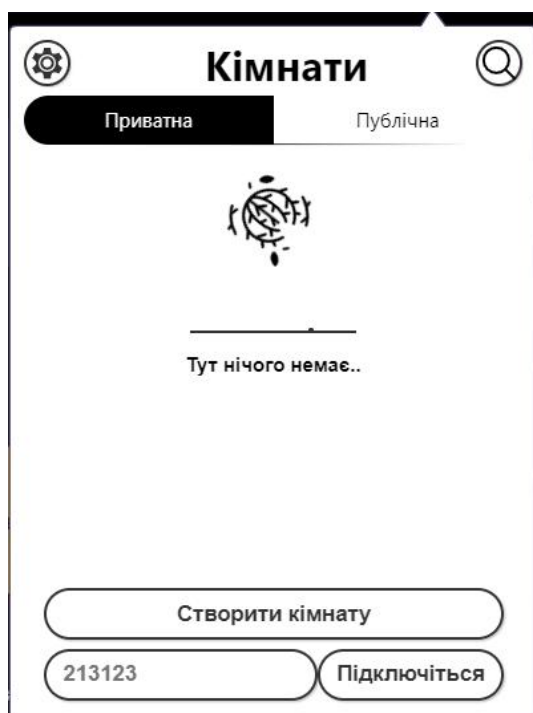


Рис. 3.4 - Головне меню

Воно має мінімалістичний дизайн та гарні анімації, що робить взаємодію з розширенням приємним та комфортним. Верхня панель має три елементи: кнопка з параметрами, назва сторінки, пошук. Натиснувши на останню кнопку, відкривається панель пошуку з параметрами:

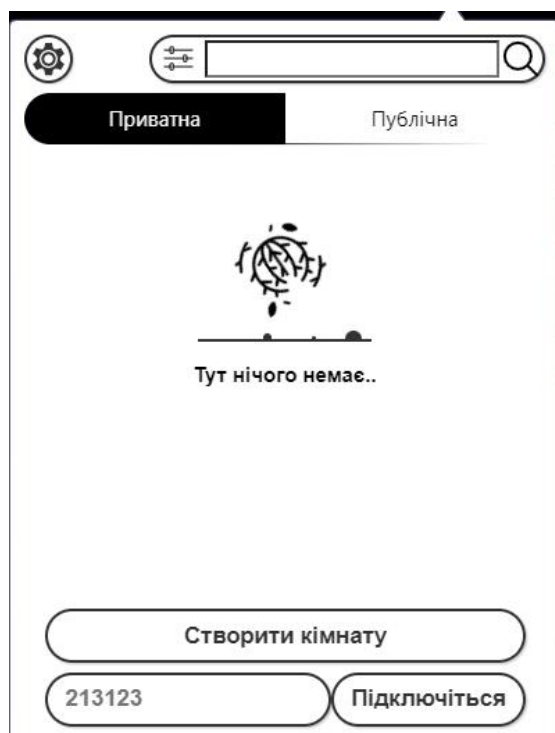


Рис. 3.5 - Панель пошуку

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

Нижче перемикач списків кімнат. Приватні кімнати - це ті, в яких знаходиться користувач, в публічному списку знаходяться створені кімнати іншими користувачами. Кімнати в списку виглядають так:



Рис. 3.6 - Кімнати в списках

Якщо користувач бажає створити кімнату, то він побачить таке меню:

Рис. 3.7 - Форма для створення кімнат

Для створення потрібно ввести ім'я кімнати та, якщо потрібно, пароль. Також можна змінити приватність на «публічно», щоб кімната з'явилась у списку для інших користувачів. Посилання та ID створюються автоматично. Якщо відкритий сайт не підтримується, то користувач побачить відповідне повідомлення. Після створення кімнати буде показана інформація про неї.

Кімната має назву, список користувачів онлайн та кнопку відкрити. Натиснувши на цю кнопку, користувач побачить інформацію про кімнату:

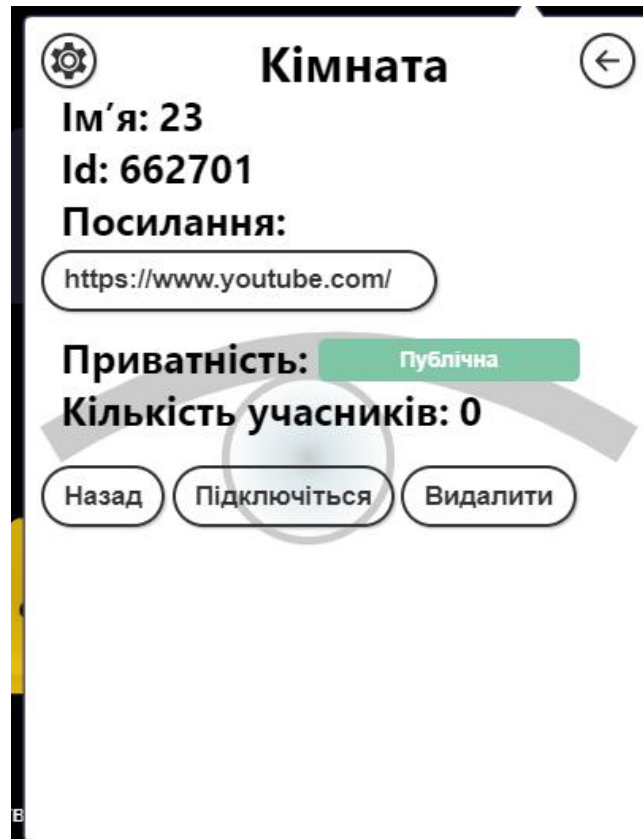


Рис. 3.8 - Інформація про кімнату

На цій сторінці є: назва кімнати, id (за яким можна знайти), посилання, за яким відбувається відкриття кімнати, приватність, кількість учасників, що зараз знаходяться там та кнопки управління. «Назад» повертає в головне меню, «підключення» перекидає в кімнату. Кнопка «Видалити» спершу запитає підтвердження, вона видаляє кімнату зі списку користувача.

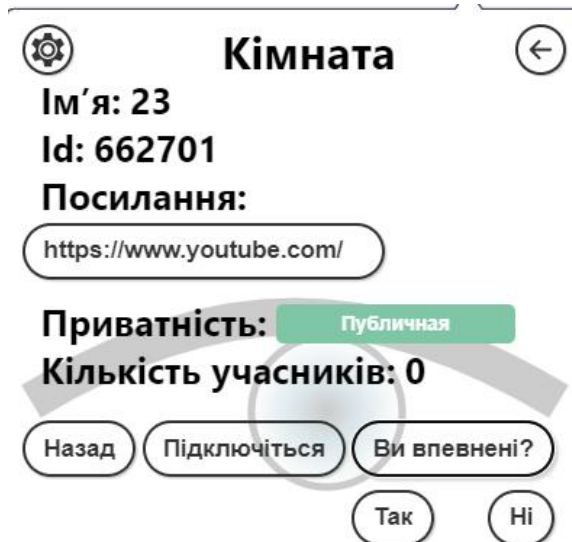


Рис. 3.9 - Підтвердження видалення

Повертаємось до головного меню. Внизу є можливість під'єднатися до кімнати, використовуючи ID. Якщо кімната має пароль, то з'явиться відповідне повідомлення:



Рис. 3.10 - Вхід з паролем

При підключенні, користувач переходить до інформації кімнати (Рис 3.1.7), кнопка «Назад» повертає форму для введення ID кімнати. При введенні неправильного пароля висвічується відповідне повідомлення:

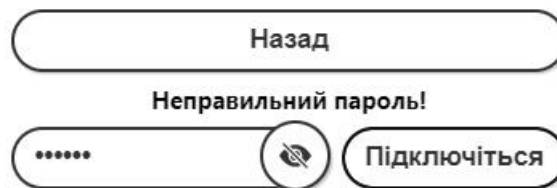


Рис. 3.11 - Повідомлення про неправильно введені дані

Перейдемо до налаштувань. При натисканні на кнопку, користувач бачить ось таку сторінку:

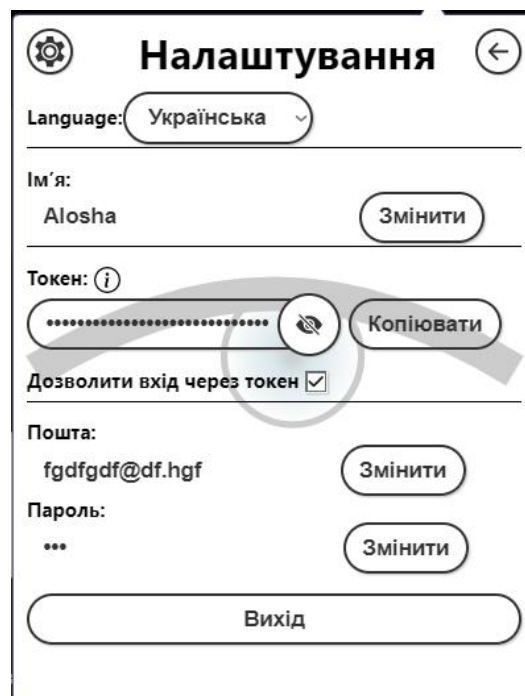


Рис. 3.12 - Налаштування

На цій сторінці можна змінити мову інтерфейсу та дані свого профілю. Також є кнопка виходу. Вона перекидає користувача на форму реєстрації (Рис. 3.2). Якщо користувач ще не вводив пошту, то у нього буде відповідна кнопка:

Налаштування

Language: Українська

Ім'я: fresh water moccasin Змінити

Токен: i Копіювати

Дозволити вхід через токен ☒

Створити пароль для входу

Вихід

Рис. 3.13 - Додати пошту та пароль

Натиснувши на неї, користувач побачить наступне:

Налаштування

Language: Українська

Ім'я: fresh water moccasin Змінити

Токен: i Копіювати

Дозволити вхід через токен ☒

Пошта: 4eyes@mail.com

Пароль: MyStrongPass

Зберегти

Вихід

Рис. 3.14 - Введення пошти та паролю

При цьому введення неправильних даних перевіряється:

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						45
Змн.	Арк.	№ докум.	Підпис	Дата		

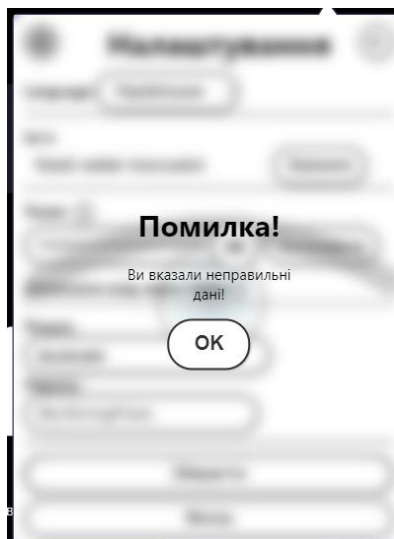


Рис. 3.15 - Неправильне введення пошти

При переході до кімнати розширення додає інтерфейс прямо на сайт. Він виглядає ось так:

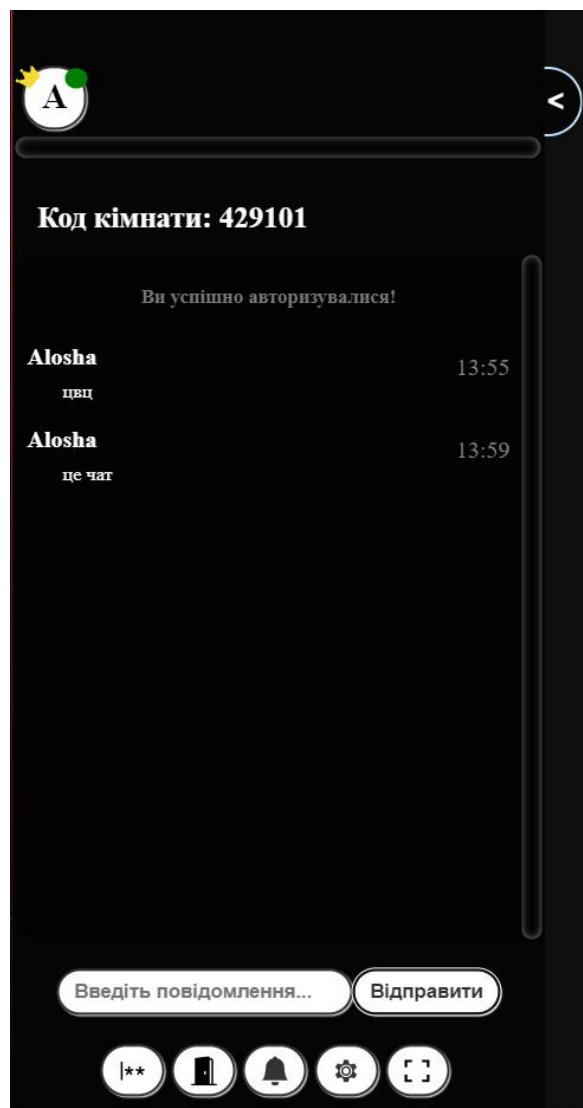


Рис. 3.16 - Інтерфейс на сайті

У цьому вікні можна побачити спосіб наявних користувачів, код кімнати, чат та панель управління. Праворуч є кнопка для згортання цього вікна, щоб воно не заважало перегляду. Усі елементи на формі мають текстові підказки при наведенні, тому зрозуміти їх призначення не викличе труднощів.

Програма має «гарячі клавіші» для контролювання розширенням. При введенні повідомлення, його можна відправити натиснувши «Enter». Щоб згорнути вікно, потрібно натиснути `ctrl + shift + A`. В такому режимі вікно ховається за межі екрану, залишивши кнопку для розгортання, ця ж кнопка є індикатором нових повідомлень:

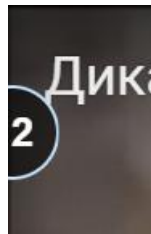


Рис. 3.17 - Індикатор нових повідомлень

Також, щоб нічого не пропустити програється звук, який можна вимкнути в панелі керування. В цій панелі перша кнопка показує пароль. Якщо пароля немає, то відобразиться відповідний текст:

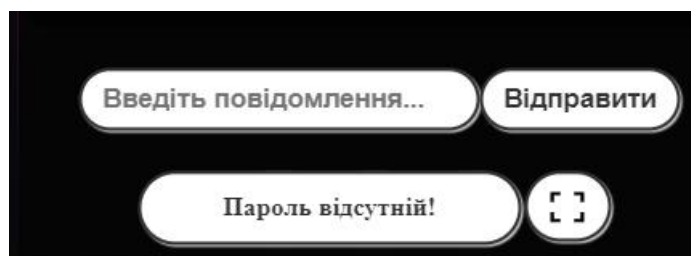


Рис. 3.18 - Пароль відсутній

Наступна кнопка - кнопка виходу, яка завершує сеанс підключення. Третя кнопка вимикає звук повідомлень:

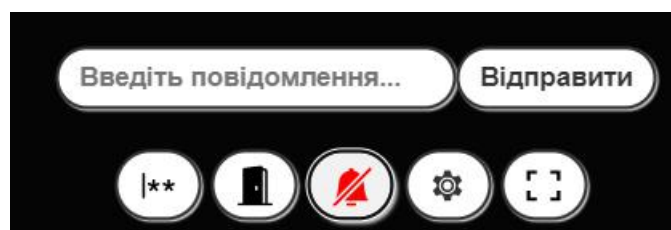


Рис. 3.19 - Вимкнені повідомлення

Остання кнопка розгортає чат на всю висоту, забираючи усе зайве. При цьому є кнопка для повернення у початковий стан:

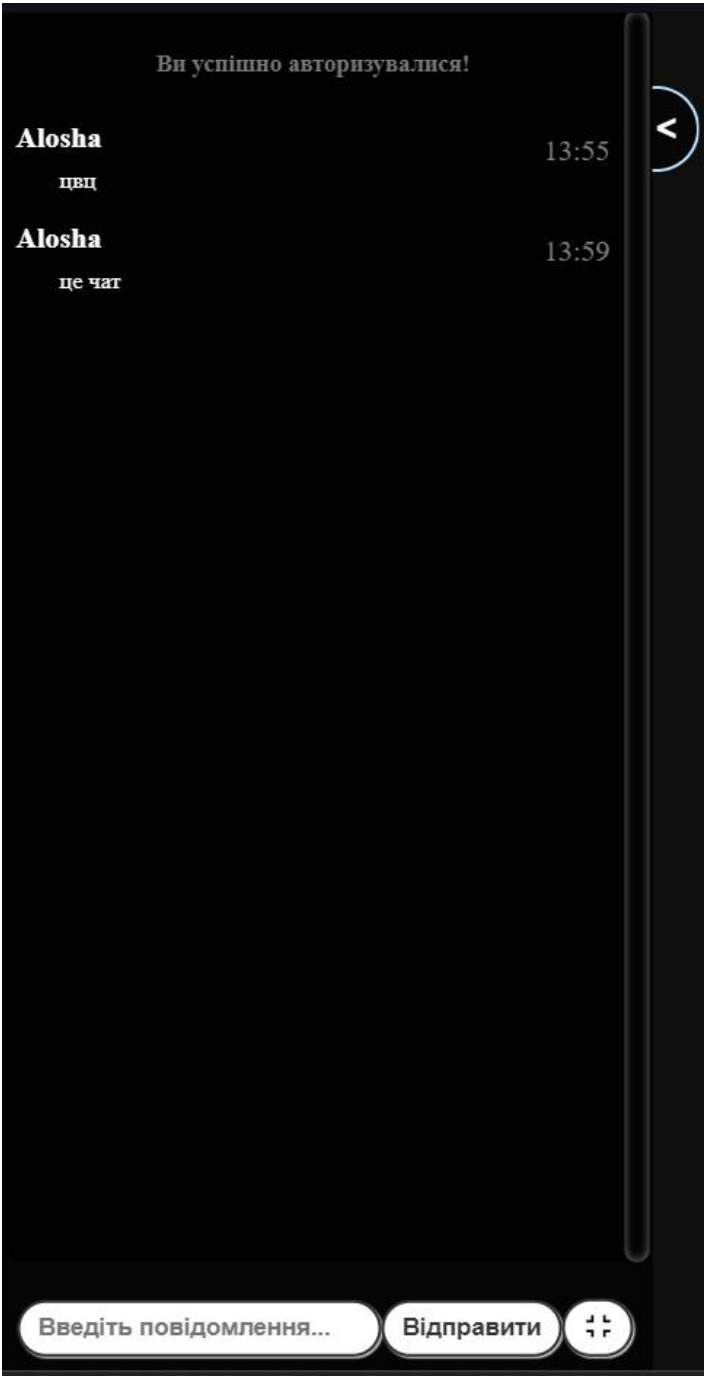


Рис. 3.20 - Чат на всю висоту

Задній фон є напівпрозорим, що дозволяє переглядати контент навіть з увімкненим чатом.

Список користувачів кімнати показані у верхній частині вікна. У кожного користувача є індикатор, що показує в якому стані у нього програвач зараз. Наприклад, сірий індикатор показує, що програвач на паузі:



Рис. 3.21 - Користувачі кімнати на паузі

Корона показує адміністратора. У нього є додатковий функціонал.

При натисканні на користувача відкривається меню:

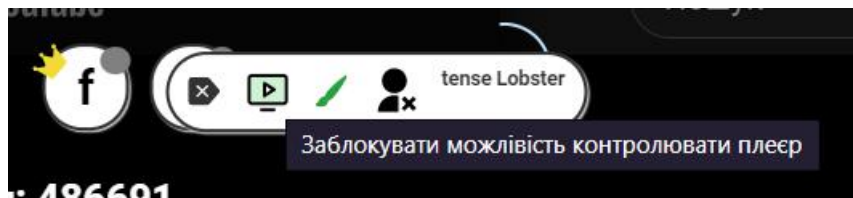


Рис. 3.22 - Управління користувачами

Тут можна заблокувати функціонал для окремого користувача або ж вигнати його взагалі. При натисканні адміністратором на свою іконку, відкривається таке ж меню, де можна заблокувати можливості для усіх користувачів відразу.

3.2 Тестування роботи програмного забезпечення

Тестування є невід'ємною частиною процесу розробки програмного забезпечення і відіграє важливу роль у забезпеченні якості та надійності продукту. Метою тестування є перевірка функціональності, стабільності та відповідності вимогам розробленого програмного забезпечення. Тому було проведено вичерпне тестування продукту для виявлення та усунення недоліків.

Таблиця 3.1

Сторінка	Вхідні дані	Очікувані дані	Спостережені дані
Реєстрації	Натискання на перемикач мов	Інтерфейс змінює весь текст розширення	Інтерфейс змінює весь текст розширення
	Натискання на кнопку генерації імен	З кожним новим натисканням у полі для імені	З кожним новим натисканням у полі для імені

		з'являється новий результат введеного тексту	з'являється новий результат введеного тексту
	Натискання на кнопку «Зареєструватися»	Розширення переходить до головного екрану маючи зареєстрований профіль	Розширення переходить до головного екрану маючи зареєстрований профіль тільки після другого натискання
	Натискання на кнопку «Увійти»	Відкривається сторінка для авторизації	Відкривається сторінка для авторизації
Авторизації	Натискання на кнопку «Назад»	Відкривається сторінка реєстрації	Відкривається сторінка реєстрації
	Натискання на кнопку «показати пароль»	У відповідному полі прибирається приховування введеного тексту, а іконка кнопки змінює свій вигляд	У відповідному полі прибирається приховування введеного тексту, а іконка кнопки змінює свій вигляд
	Натискання кнопки входу з існуючими даними	Розширення переходить до головного екрану маючи існуючий профіль	Розширення переходить до головного екрану маючи існуючий профіль
	Спроба увійти з	Відкривається	Відкривається

	некоректними, неіснуючими даними	повідомлення про помилку	повідомлення про помилку
	Спроба входу з існуючим токеном	Розширення переходить до головного екрану маючи існуючий профіль	Розширення переходить до головного екрану маючи існуючий профіль
Головне меню	Натискання на вибір списку кімнат	Кнопка візуально змінюється. Список кімнат оновлюється	Кнопка візуально змінюється. Список кімнат оновлюється
	Натискання на кнопку налаштувань	Відкривається вікно налаштувань	Відкривається вікно налаштувань
	Підключення до кімнати, що має пароль	Панель вводу ID кімнати замінюється на форму з веденням паролю	Панель вводу ID кімнати замінюється на форму з веденням паролю
	Натискання на кнопку «відкрити кімнату»	Відкривається сторінка з інформацією про кімнати	Відкривається сторінка з інформацією про кімнати
	Натискання на кнопку пошуку	Відкривається панель для пошуку	Відкривається панель для пошуку

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						51
Змн.	Арк.	№ докум.	Підпис	Дата		

	Натискання на кнопку «Створити кімнату»	Якщо сайт підтримує, відкривається сторінка створення кімнати, інакше - повідомлення про помилку	Усі варіанти виконуються згідно з очікуваннями
Налаштування	Натискання на перемикач мов	Інтерфейс змінює весь текст розширення	Інтерфейс змінює весь текст розширення
	Натискання на кнопки «Змінити»	Відповідне поле для введення стає доступним для редагування. Кнопка змінює свій текст	Відповідне поле для введення стає доступним для редагування. Кнопка змінює свій текст
	Натискання на кнопку «показати пароль»	У відповідному полі прибирається приховування введеного тексту, а іконка кнопки змінює свій вигляд	У відповідному полі прибирається приховування введеного тексту, а іконка кнопки змінює свій вигляд
	Натискання кнопки створення паролю	Відкривається панель з введенням пошти	Відкривається панель з введенням пошти

		та пароллю. Кнопка змінює свій текст	та пароллю. Кнопка змінює свій текст
	Натискання кнопки «Вихід»	Вихід з профілю та відкриття сторінки реєстрації	Вихід з профілю та відкриття сторінки реєстрації
Інформація про кімнату	Натискання на кнопку «Назад»	Відкривається головна сторінка	Відкривається головна сторінка
	Натискання на кнопку «Підключитись»	Відкривається сторінка, що вказана в посиланні та підключаються скрипти	Відкривається сторінка, що вказана в посиланні та підключаються скрипти
	Натискання на кнопку «Видалити»	Текст кнопки змінюється підтвердження та з'являються додаткові кнопки вибору	Текст кнопки змінюється підтвердження та з'являються додаткові кнопки вибору
	Натискання на кнопку підтвердження або на кнопку «Ні»	Повертає кнопку видалення у початковий стан	Повертає кнопку видалення у початковий стан
	Натискання на кнопку «Так» при підтвердженні	Відкривається головне меню. Кімната зникає зі	Відкривається головне меню. Кімната зникає зі

	видалення кімнати	списку користувацьких кімнат	списку користувацьких кімнат
Створення кімнати	Спроба створити кімнату без назви	Відкривається повідомлення про помилку	Створюється кімната без назви
	Натискання на кнопку «показати пароль»	У відповідному полі прибирається приховування введеного тексту, а іконка кнопки змінює свій вигляд	У відповідному полі прибирається приховування введеного тексту, а іконка кнопки змінює свій вигляд
	Натискання на параметр приватності кімнати	Блок анімовано змінює своє значення та колір	Блок анімовано змінює своє значення та колір
	Натискання на кнопку «Назад»	Відкривається головна сторінка	Відкривається головна сторінка
	Натискання на кнопку «Створити кімнату»	Відкривається інформація про створену кімнату	Відкривається інформація про створену кімнату

При тестуванні були виявлені помилки, незручності та недоліки розширення. Наприклад, підключення на сторінці реєстрації відбувається після спроби відправити запит. Це відбувається через асинхронне підключення WebSocket. Це можна виправити через Promise. Також було помічено, що деякі елементи при відпрацювання не повертаються у початковий стан, тому їх потрібно додати у спеціально створений метод. При створенні нової кімнати потрібно додати перевірку імені кімнати, щоб не допускати пуского поля.

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновки до третього розділу

В третьому розділі було сформульовано детальний опис роботи з програмним додатком та його тестування. Було наведено скріншоти візуальної частини продукту.

Було проведено аналіз роботи та вичерпне тестування програмного забезпечення з метою виявлення недоліків та незручностей.

Спираючись на проведений аналіз, було обрано шляхи вирішення виявлених проблем та недоліків, проведено тестування та внесено необхідні зміни для поліпшення роботи програмного забезпечення.

В результаті було досягнуто більшої якості та відповідності вимогам розробленого продукту.

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У рамках даної роботи було розроблено сервіс для одночасного перегляду відеоконтенту віддаленими користувачами.

Для побудови сервісу використано клієнт-серверну архітектуру та такі технології як WebSockets, JavaScript, HTML, CSS, C# та Entity Framework для бази даних було використано PostgreSQL, що надає зручний спосіб спільного перегляду фільмів, телешоу та іншого відеоконтенту незалежно від розташування користувачів.

В першому розділі було проаналізовано проблематику та існуючі аналоги, такі як Netflix Party, Kast, TwoSeven, що підтвердило актуальність та популярність подібних сервісів.

В другому розділі було спроектовано та створено архітектуру системи, яка враховує вимоги та потреби користувачів.

Для забезпечення швидкого та надійного з'єднання клієнтів з сервером було використано WebSockets, який дозволяє забезпечити в режимі реального часу швидку передачу даних між користувачами. Встановлення з'єднання по протоколу WSS забезпечує безпечну комунікацію між клієнтом та сервером.

Також було розроблено базу даних, де використано Entity Framework для забезпечення зручного доступу та керування даними.

В третьому розділі було детально описано користувацький інтерфейс зі скріншотами.

Під час тестування роботи програмного забезпечення були проведені різні види тестів, що дозволило перевірити функціональність, стабільність та відповідність вимогам розробленого сервісу. Цей процес допоміг виявити та усунути можливі помилки та проблеми для покращення якості та надійності продукту.

У майбутньому можливе розширення функціональності сервісу. Можна додати нові можливості, такі як оцінювання контенту, можливість створення плейлистів, голосовий чат тощо, щоб зробити його ще більш привабливим та функціональним для користувачів. Також, необхідно забезпечити

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						56
Змн.	Арк.	№ докум.	Підпис	Дата		

масштабованість та стабільність системи при зростанні кількості користувачів та навантаження на сервер.

Було розроблено сервіс, який надає можливість одночасного перегляду одного контенту декільком користувачам, що знаходяться далеко один від одного. Це досягнуто завдяки використанню технологій інтерактивного спілкування, зокрема протоколу WebSockets, який забезпечує швидкий та безперебійний обмін даними між клієнтами та сервером.

Даний сервіс здатний створити відчуття присутності іншої людини, навіть у випадках, коли вона знаходиться на значній відстані. Спільний перегляд контенту сприяє покращенню комунікації та взаємодії між людьми, надаючи їм можливість спільно насолоджуватися фільмами, телешоу, відеопрезентаціями та іншим відеоконтентом. Цей сервіс забезпечить унікальний досвід, дозволивши користувачам віддалено переглядати відеоконтент на своїх улюблених платформах разом з іншими користувачами.

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						57
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. U.S.: film viewing in theaters versus at home 2022 | Statista [Електронний ресурс] // Statista. – 2022. – Режим доступу до ресурсу: <https://www.statista.com/statistics/1330643/watch-movie-theaters-home-united-states/>.
2. Netflix Party is now Teleparty. Google Chrome - Download the Fast, Secure Browser from Google. [Електронний ресурс] – Режим доступу до ресурсу: <https://chrome.google.com/webstore/detail/netflix-party-is-now-tele/oocalimimngaihdkbihfgmpkcpnmlaoa>.
3. Kast is Watch Parties Made Easy. Kast is Watch Parties Made Easy. [Електронний ресурс] – Режим доступу до ресурсу: <https://kastapp.co>.
4. Watch2Gether - Watch Videos together. [Електронний ресурс] // Watch2Gether – Режим доступу до ресурсу: <https://w2g.tv/en/>.
5. H J. How HTML, CSS, and JavaScript work [Електронний ресурс] / H J. // Scrimba blog – Режим доступу до ресурсу: <https://scrimba.com/articles/html-css-javascript/>.
6. A tour of C# - Overview. Microsoft Learn: Build skills that open doors in your career. [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/>.
7. Advantages of Websockets | disadvantages of Websockets. RF Wireless Vendors and Resources | RF Wireless World. [Електронний ресурс] – Режим доступу до ресурсу: <https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-Websockets.html>.
8. Laukkonen J. What is Chromium Browser Used For? [Електронний ресурс] / Laukkonen J. // Lifewire – Режим доступу до ресурсу: <https://www.lifewire.com/chromium-web-browser-4171288>.
9. Smith L. What postgresql has over other open source sql databases [Електронний ресурс] / Lisa Smith – Режим доступу до ресурсу: <https://www.compose.io/articles/what-postgresql-has-over-other-open-source-sql-databases/>.

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

10. What is Entity Framework?. Entity Framework Tutorial. [Електронний ресурс] – Режим доступу до ресурсу: <https://www.entityframeworktutorial.net/what-is-entityframework.aspx>.
11. What is Client Server Architecture? - Differences, Types, Example [Електронний ресурс] // Intellipaat Blog – Режим доступу до ресурсу: <https://intellipaat.com/blog/what-is-client-server-architecture/?US>.
12. What is web socket and how it is different from the HTTP? [Електронний ресурс] // GeeksforGeeks – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/what-is-web-socket-and-how-it-is-different-from-the-http/>.
13. Let's Encrypt - Free SSL/TLS Certificates [Електронний ресурс] // Let's Encrypt – Режим доступу до ресурсу: <https://letsencrypt.org/uk/>.
14. WebSocket - Web APIs | MDN [Електронний ресурс] // MDN Web Docs – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/API/WebSocket>.
15. CodePen Home Bouncing Tumbleweed Animation [Електронний ресурс] – Режим доступу до ресурсу: <https://codepen.io/aido179-1471800764/pen/jZJjpo>.
16. Video and audio content - Learn web development | MDN [Електронний ресурс] // MDN Web Docs – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Learn/HTML/Multimedia_and_embedding/Video_and_audio_content.
17. animation - CSS: Cascading Style Sheets | MDN [Електронний ресурс] // MDN Web Docs – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/CSS/animation>.
18. Price M. J. C# 7 and .NET Core: Modern Cross-Platform Development: Create powerful cross-platform applications using C# 7, .NET Core, and Visual Studio 2017 or Visual Studio Code, 2nd Edition. Packt Publishing, 2017. - 594 с.
19. Томашевський В.В. Технології програмування. Мова С# : навч. посібн. – Житомир: ЖВІ НАУ, 2012. – 484 с.

					КН.КР.Б – 122 – 13 – ПЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

20. Chrome Extensions getting started guides - Chrome Developers [Електронний ресурс] // Chrome Developers – Режим доступу до ресурсу: <https://developer.chrome.com/docs/extensions/mv3/getstarted/>.
21. Svg Icons [Електронний ресурс] – Режим доступу до ресурсу: <https://fontawesomeicons.com/svg/icons>.
22. Kosaka M. Same-origin policy [Електронний ресурс] / Mariko Kosaka – Режим доступу до ресурсу: <https://web.dev/same-origin-policy/#:~:text=The%20same-origin%20policy%20is,from%20multiple%20sites%20at%20once>.
23. WHAT IS AN SSL CERTIFICATE? [Електронний ресурс] – Режим доступу до ресурсу: [https://www.digicert.com/what-is-an-ssl-certificate#:~:text=Secure%20Sockets%20Layer%20\(SSL\)%20is,client%20\(e.g.%2C%20Outlook\)](https://www.digicert.com/what-is-an-ssl-certificate#:~:text=Secure%20Sockets%20Layer%20(SSL)%20is,client%20(e.g.%2C%20Outlook)).
24. Create and publish custom Chrome apps & extensions [Електронний ресурс] – Режим доступу до ресурсу: <https://support.google.com/chrome/answer/2714278?hl=en&sjid=6423533847990866837-EU>.
25. pseudo-elements ::after [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/CSS/::after>.

ДОДАТКИ

Найбільш важливі функції

PageScript.js:

```

function Connect(){
    let room;
    storage.get("eyesId",function(items){
        hash = items.eyesId;
        storage.get("eyesRoom",function(items){
            room = items.eyesRoom;
            try {
                websocket = new WebSocket
( `${url}?hash=${hash}&roomId=${room}` );
                websocket.onmessage = function (event) {

                    const obj = JSON.parse(event.data);
                    switch (obj.code){
                        case -1: // error
                            alert(obj.data);
                            break;
                        case 0: //connect
                            let data = JSON.parse(obj.data);
                            console.log(data)
                            console.log(data.ctime)
                            if(localStorage.getItem('eyesId') === null){
                                console.log(data.newID);
                                localStorage.setItem('eyesId', data.eyesID);
                                console.log('получил новый ид');
                            }else {
                                console.log('id уже был ' + data.eyesID);
                            }
                            //eyesPlayer.currentTime = data.ctime;
                            //UpdatePlayerEvent();
                            myId = data.ID;
                            adminID = data.admin;
                            console.log(`me:${myId}\nadmin:${adminID}`)
                            CreateChat(data);
                            //UpdateUserList(data.userList, myId, obj.status)
                            break;
                        case 1:{
                            if(obj.msg === "true"){
                                newMessageFromServer(obj.data);
                            }
                            adminID = obj.admin;
                            UpdateUserList(obj.name, myId, obj.status)
                            break;
                        }
                        case 3: //chat
                            newMessageFromServer(obj.data, obj.name);
                            break;
                        case 4://Player
                            let info = JSON.parse(obj.data)
                            console.log(info)
                            switch (info.type){
                                case 'play':

```

```

        if(info.value === true){
            ChangeDuration(info.time);
            Play();
        }

//eyesPlayer.dispatchEvent(readyForPlayEvent);
    }else {
        prevCurrentTime = eyesPlayer.currentTime;
        Pause();
    }
    break;
case 'seeked':
    ChangeDuration(info.time);
    break;
case 'rate':
    console.log('смена скорости')
    if(eyesPlayer.playbackRate !== info.value){
        console.log('поменял')
        eyesPlayer.playbackRate = info.value;
    }
    break;
case 'volumechange':
    //доделать
    //eyesPlayer.volume = info.value;
    break;
case 'alert':{
    break;
}
}
break;
case 6:
    if(obj.data === "kick"){
        document.getElementById('ExitB').click();
        alert(lang['Kicked']);
    }
    break;
case 9:{
    console.log('wanna play')
    if (eyesPlayer.readyState === 4){
        console.log('отправляю что готов включить')
        SentPlayerInfo('play', true, 9);
        eyesPlayer.onreadystatechange = null;
    }else{
        eyesPlayer.onreadystatechange = function (){
            if(eyesPlayer.readyState === 4){
                console.log('Дождался готовности и
отправляю что готов включить')
                SentPlayerInfo('play', true, 9);
                eyesPlayer.onreadystatechange = null;
            }
        }
    }
    break;
}
case 10:
{
    let info = JSON.parse(obj.data)

```

```

        let userId = 'eyes-user-' + info.id;
        console.log(userId);
        let frame = document.getElementById('eyesFrame');

        let user =
frame.contentWindow.document.getElementById(userId).children[0];
        if(!info.value){
            user.classList.add('paused');
        }else {
            user.classList.remove('paused');
        }
    }
}

};

websocket.onerror = function (event) {
    newMessageFromServer (lang['Error'] + event.data);
};

websocket.onclose = function (event) {
    if(attReconnect > 5) return;
    if(!event.wasClean){
        newMessageFromServer (lang['Disconnect']);
        init(code,name, '',true);
        attReconnect++;
        setTimeout(() => { }, 5000);
    }else{
        document.getElementById('output')?.remove();
    }
};
}catch{
    let frame = document.getElementById('eyesFrame');
    frame.contentWindow.document.getElementsByClassName('eyes-
content')[0].innerHTML = lang['ErrorConnect'];
}
});
});
}

function newMessageFromServer(message, name){
    let frame = document.getElementById('eyesFrame');
    let output = frame.contentWindow.document.getElementById ("eyesChat");
    if(!name){
        var messageP = document.createElement ("p");
        messageP.innerText = message;
        output.appendChild (messageP);
    }else {
        let messageDiv = document.createElement ("div");
        messageDiv.className = 'eyesMessage';
        let messageName = document.createElement ("div");
        messageName.className = 'eyesNameUser';
        messageName.innerText = name;
        messageDiv.append(messageName);
        let timeDiv = document.createElement('div');
        timeDiv.className = 'timeDiv';
        let ctime = new Date();
        timeDiv.innerText = ctime.getHours().toString().padStart(2, '0')
+ ':' + ctime.getMinutes().toString().padStart(2, '0');

```

```

        messageDiv.appendChild(timeDiv);
        let messageContent = document.createElement("div");
        messageContent.className = 'eyesContentMess';
        messageContent.innerText = message;
        messageDiv.appendChild(messageContent);
        output.appendChild(messageDiv);
        messageDiv.scrollIntoView({block: "center", behavior: "smooth"});
    }
    if (!isOpenPanel){
        countNewMess++;
        openPanelButton.innerText = countNewMess;
        if (enableSound){
            sound.play();
        }
    }
}
function LoadInterface(){
    let videoPlayers = [];
    let player = document.getElementsByTagName('video');
    videoPlayers.push(...player);
    let frames = document.getElementsByTagName('iframe');
    for (let i = 0; i < frames.length; i++) {
        try {
            var iframeWindow = frames[i].contentWindow;
            var iframeElement =
iframeWindow.document.getElementsByTagName('video');
            videoPlayers.push(...iframeElement);
        }catch { }
    }
    if(videoPlayers.length === 1){
        eyesPlayer = videoPlayers[0];
        UpdatePlayerEvent();
    }else if(videoPlayers.length === 0){
        alert('not good')
    }else{
        alert('not good but trying fix')
    }
    let frame = document.createElement('iframe');
    frame.id = 'eyesFrame';
    frame.style = 'z-index: 2147483647; position: fixed; width: 100%;
height: 100%; top: 0px; border: none; clip-path: polygon(0px 0px, 363px
0px, 363px 38px, 387px 49px, 387px 76px, 363px 97px, 363px 100%, 0px 100%);
transition: all 280ms ease-in-out 0s; left: 0px;'
    let mak = document.createElement('div');
    mak.innerHTML = `
        <div style="
height: 100vh;
width: 360px;
background: #6624a1;
/* z-index: 999999999999; */
position: absolute;
top: 0;
"></div>
        <div></div>
    `
    mak.onmouseover = function (){
        console.log('over')
    }
}

```

```

    }
    mak.onmouseout = function () {
        console.log('out');
    }
    document.body.insertBefore(frame, document.body.firstChild);
    document.body.insertBefore(mak, document.body.firstChild);
    let div = document.createElement('div');
    //div.innerHTML = GetTable();
    div.id = 'output'
    //document.body.append(div)
    //main.appendChild(div);
    div.innerHTML = GetTable();
    frame.contentWindow.document.body.append(div);
    frame.contentWindow.document.head.appendChild(styleElement);
    openPanelButton =
frame.contentWindow.document.getElementById('openPanel');
    openPanelButton.onclick = function () {
        if(isOpenPanel){
            this.innerHTML = '>';
            frame.style.left = '-360px';
            isOpenPanel = false;
        }else {
            countNewMess = 0;
            this.innerHTML = '<';
            frame.style.left = '0';
            isOpenPanel = true;
        }
    }
}

```

Popup.js:

```

function injectScript() {
    let info = document.getElementById('RoomInfo');
    let urlR = info.children[2].children[0].value;
    storage.set({
        eyesRoom:info.children[1].innerText
    },function () {
        chrome.tabs.query({active: true, currentWindow: true}, (tabs) => {
            console.log(urlR)
            const tab = tabs[0];
            if (tab.url === urlR) {
                chrome.scripting.executeScript({
                    target: {tabId: tab.id},
                    files: ["Player.js", "Vocabulary.js", "pageScript.js"],
                });
            } else {
                chrome.tabs.create({url: urlR}, async function (tab) {
                    await chrome.tabs.update(tab.id, {active: true});
                    await chrome.scripting.executeScript({
                        target: {tabId: tab.id},
                        files: ["Player.js", "Vocabulary.js", "pageScript.js"],
                    });
                });
            }
        })
    })
}

```

```

    });
}
function FindPlayer(){
    let videoPlayers = [];
    let player = document.getElementsByTagName('video');
    videoPlayers.push(...player);
    let frames = document.getElementsByTagName('iframe');
    let url = document.location.href;
    for (let i = 0; i < frames.length; i++) {
        try {
            var iframeWindow = frames[i].contentWindow;
            var iframeElement =
iframeWindow.document.getElementsByTagName('video');
            videoPlayers.push(...iframeElement);
        }catch {

        }
    }
    console.log(videoPlayers);
    return {
        count: videoPlayers.length,
        url: url
    };
}
function ActiveTab(tab){
    if ([0,4,5,7].includes(tab)){
        document.getElementById('Header').classList.add('vis');
    }else{
        document.getElementById('Header').classList.remove('vis');
    }
    let children = document.getElementById('main').children;
    for (let i = 0; i < children.length; i++) {
        children[i].classList.remove('active');
    }
    children[tab].classList.add('active');
    document.getElementById('pageName').innerText =
lang[children[tab].dataset.name];
    let back = document.getElementById('backButton');
    let find = document.getElementById('FilterButton');
    if(tab === 1){
        back.style.display = 'none';
        find.style.display = 'flex';
    }else {
        find.style.display = 'none';
        back.style.display = 'flex';
    }
    let a = document.getElementById('deleteRoom');
    if (a.innerText !== lang['Delete']){
        a.click();
    }
    document.getElementById('roomId').value = '';
    document.getElementById('SearchAndFilter').classList.remove('max');
}
function Connect(){
    storage.get("eyesId",function(items){
        let hash = items.eyesId;
        if(!items.eyesId) hash = 'new';
    });
}

```

```

websocket = new WebSocket(url + 'connect?hash=' + hash);
Loading(document.getElementById('rooms'));
websocket.onmessage = function (event) {
    const obj = JSON.parse(event.data);
    console.log('code:' + obj.code);
    switch (obj.code){
        case -1: // error
            switch (obj.data){
                case 'reconnect':{
GenerateErrorPage(lang['ReConnectDone'],'OK',function (){
                    ActiveTab(PageTab.loading);
                });
                break;
            }
            case 'mailok':{
                break;
            }
            default:
                GenerateErrorInfo(Error, lang[obj.data]);
                /*GenerateErrorPage(obj.data, 'OK', function (){
                    ActiveTab(PageTab.main);
                });*/
                break;
        }
        console.log('case -1')
        break;
    case 0:
        console.log('case 0')
        console.log(obj);
        let userInfo = JSON.parse(obj.data);
        if(userInfo.status === 'OK'){
            EnterUserInformation(userInfo);
            if(!haveRooms)
                send(0, '0');
        }
        else if(userInfo.status === 'Re-entry'){
            document.getElementById('errorMessage').innerText =
lang['ReConnectError'];
            ActiveTab(PageTab.error);
GenerateErrorPage(lang['ReConnectError'], lang['ReConnect'], SetNewConnect);
        }
        break;
    case 1://rooms
        console.log();
        haveRooms = true;
        reconnectCount = 0;
        console.log('case 1')
        let info = JSON.parse(obj.data)
        console.log(info);
        ActiveTab(PageTab.main);
        let userRooms = document.getElementById('rooms');
        userRooms.innerHTML = '';
        let publicRooms = document.getElementById('publicRooms');

```

```

publicRooms.innerHTML = '';
for (let i = 0; i < info.length; i++){
    console.log(info[i].isPublic === true);
    if(info[i].userRoom){
        console.log('dsfe')
        AddToListRoom(info[i], userRooms);
    }else if(info[i].isPublic === true){
        console.log('sdfsd')
        AddToListRoom(info[i], publicRooms);
    }
}
GeneratePagingButtons(obj.count, obj.page);
paginationCash[obj.page] = true;
paginationCash[obj.page] = publicRooms.innerHTML;
if(userRooms.innerHTML === ''){
    console.log('yayaya')
    EmptyPage(userRooms);
}

if(publicRooms.innerHTML === ''){
    console.log('yoyoyo')
    EmptyPage(publicRooms);
}
break;
case 2:
    let roomInfo = JSON.parse(obj.data)
    let block = document.getElementById('RoomInfo');
    block.children[0].innerText = roomInfo.name;
    block.children[1].innerText = roomInfo.id;
    block.children[2].children[0].value = roomInfo.link;
    block.children[3].children[0].checked = roomInfo.isPublic;
    block.children[4].innerText = roomInfo.count;
    ActiveTab(PageTab.roomInfo);
    if(!document.querySelector('#rooms [data-
id="${roomInfo.id}"]`'))
        AddToListRoom(roomInfo,
document.getElementById('rooms'));
    break;
case 3:
    alert('+++')
    break;
case 4:
    console.log(obj.data)
    if(obj.data === 'true'){
        PasswordPanel();
    }
    else {
        document.getElementById('passInfo').innerText =
lang['WrongPass'];
    }
    break;
case 5:
    let room = JSON.parse(obj.data)
    if(obj.data.existence){

    }else{
        let id = room.roomId;

```

```

        let m = document.getElementById('rooms');
        document.querySelector(`#rooms [data-
id="${id}"]`).remove();
        if(m.innerText === ''){
            EmptyPage(m);
        }
    }
    break;
case 6:
    break;
}
};
websocket.onerror = function (event) {
    GenerateErrorPage(lang['ErrorConnect'], lang['ReConnect'],
Reconnect);
};

websocket.onclose = function (event) {
    if(reconnectCount > 5) return;
    reconnectCount++;
    Reconnect();
    setTimeout(() => { }, 5000);
};
});
}

```

Vocabulary.js:

```

var userLang = navigator.language || navigator.userLanguage;
var lang;

```

```

if(userLang.match("ua")) lang = SetLanguage("ua");
else lang = SetLanguage("en");

```

```

const availableLanguages = [
    { code: 'ua', name: 'Українська' },
    { code: 'en', name: 'English' },];

```

```

function SetLanguage(lang){
    switch (lang) {
        case "ua":{
            return {
                "Login": "Авторизація",
                "Rooms" : "Кімнати",
                "CreateRoom": "Створити кімнату",
                "RoomInfo": "Кімната",
                "Error" : "Помилка!",
                "Settings" : "Налаштування",

                "NotPass": "Залиште пустим, якщо не бажаєте мати пароль",
                "CodeErr" : "Введіть правильний код кімнати!",
                "ErrorConnect" : "Відсутнє підключення до сервера",
                "ErrorData": "Ви вказали неправильні дані!",
                "Disconnect" : "Ви відключені!",
            }
        }
    }
}

```

```
"Exit" : 'Вихід',
"Yes" : 'Так',
"No" : 'Ні',
"fullChat": 'Чат на всю висоту',
"ExistRoom": 'до існуючої кімнати',
"Connect": 'Підключіться',
>Delete": 'Видалити',
>DeleteConf": 'Ви впевнені?',
"OrByCode": 'або підключіться за кодом до існуючої кімнати',
>CreateUser": 'Створити користувача',
"ThenClick": 'а потім натисніть',
"FirstNick": 'Спочатку придумайте собі нік',
"value" : "ua",
"Back" : "Назад",
"BlockPlay" : "Заблокувати можливість контролювати плеєр",
"BlockPaint" : "Заблокувати можливість малювати",
"ConnectWithoutP" : "Без пароля",
"MPass" : "Можливо, додати пароль?",
"RPass" : "Ця кімната має пароль!",
"Kick" : "Вигнати користувача",
"Kicked" : "Адміністратор вигнав вас із кімнати!",
>ShowPass" : "Показати пароль",
"WithoutPass": "Пароль відсутній!",
>Welcome" : "Ви успішно авторизувалися!",
>Send": "Відправити",
>SendMess" : "Введіть повідомлення...",
>soundButton" : "Звукові повідомлення",
>forAll" : " для всіх користувачів",
>LoginWH" : "Бажаєте увійти з допомогою хеша?",
>Open" : "Відкрити",
>Restore" : "Відновити",
>NameRoom" : "Ім'я:",
>IdRoom" : "Id:",
>LinkRoom" : "Посилання:",
>IsPublicRoom" : "Приватність:",
>CountUserRoom" : "Кількість учасників:",
>Pass" : "Пароль:",
>Public" : "Публічна",
>Private" : "Приватна",
>ReConnect" : "Перепідключитися",
>ReConnectDone" : "Ви підключились через інший пристрій!",
>ReConnectError" : "Ви вже маєте активне підключення! Бажаєте його закрити?",
>WrongPass" : "Неправильний пароль!",
>SignIn" : "Увійти",
>SignUp" : "Зареєструватися",
>HaveAcc" : "Вже є аккаунт?",
>GenNick" : "Згенерувати новий нікнейм",
>Maybe" : "Можливо...",
>Nothing" : "Тут нічого немає..",
>CreatePass" : "Створити пароль для входу",
>Change" : "Змінити",
>HashLogin" : "Дозволити вхід через токен",
>Hash" : "Токен:",
>HashW" : "Нікому не повідомляйте свій токен!",
>Mail" : "Пошта:",
>Copy" : "Копіювати",
```

```

        "Save" : "Зберегти",
        "" : "",
    }
}

case "en":{
return {
    "Login":"Authorization",
    "Rooms" : "Rooms",
    "CreateRoom": 'Create a room',
    "RoomInfo":"Room",
    "Error": "Error!",
    "Settings" : "Settings",

    "NotPass": "Leave blank if you do not want a password",
    "CodeErr": "Please enter the correct room code!",
    "ErrorConnect": "No connection to server",
    "ErrorData": "You have entered incorrect data!",
    "Disconnect": "You are disconnected!",
    "Exit": 'Exit',
    "Yes": 'Yes',
    "No": 'No',
    "fullChat": 'Full height chat',
    "ExistRoom": 'to an existing room',
    "Connect": 'Connect',
    "Delete": 'Delete',
    "DeleteConf": 'Are you sure?',
    "OrByCode": 'or connect by code to an existing room',
    "CreateUser": 'Create user',
    "ThenClick": 'and then click',
    "FirstNick": 'First, come up with a nickname',
    "value" : "en",
    "BlockPlay" : "Block the ability to control the player",
    "BlockPaint" : "Block the ability to draw",
    "ConnectWithoutP" : "No password",
    "Back" : "Back",
    "MPass" : "Maybe add a password?",
    "RPass" : "This room has a password!",
    "Kick" : "Kick the user out",
    "Kicked" : "The administrator kicked you out of the room!",
    "ShowPass" : "Show password",
    "WithoutPass" : "The password is missing!",
    "Welcome" : "You have successfully logged in!",
    "Send" : "Send",
    "SendMess" : "Enter your message...",
    "soundButton" : "Sound notifications",
    "forAll" : " for all users",
    "LoginWH" : "Want to login with a hash?",
    "Open" : "Open",
    "Restore" : "Restore",
    "NameRoom" : "Name:",
    "IdRoom" : "Id:",
    "LinkRoom" : "Link:",
    "IsPublicRoom" : "Privacy:",
    "CountUserRoom" : "Number of participants:",
    "Pass" : "Password:",
    "Public" : "Public",
    "Private" : "Private",

```

```

        "ReConnect" : "Reconnect",
        "ReConnectDone" : "You are connected through another device!",
        "ReConnectError" : "You already have an active connection! Do
you want to close it?",
        "WrongPass" : "Wrong password!",
        "SignIn" : "Sign in",
        "SignUp" : "Sign up",
        "HaveAcc" : "Already have an account?",
        "GenNick" : "Generate new nickname",
        "Maybe" : "Maybe...",
        "Nothing" : "There's nothing here",
        "CreatePass" : "Create a login password",
        "Change" : "Change",
        "HashLogin" : "Allow login via token\n",
        "Hash" : "Token:",
        "HashW" : "Don't share your token with anyone!",
        "Mail" : "Mail:",
        "Copy" : "Copy",
        "Save" : "Save",
        "" : "",
    }
}
}
}
}

```

Program.cs:

```

using System.Net.Security;
using System.Net;
using System.Security.Authentication;
using System.Security.Cryptography.X509Certificates;
using WebSocketSharp;
using WebSocketSharp.Server;

namespace SocketServer
{
    public class Program
    {
        public static void Main(string[] args)
        {
            var wssv = new WebSocketServer(8080);
            // var wssv = new WebSocketServer(8080, true);
            string filesd = Directory.GetCurrentDirectory() + "/sert.pfx";
            Console.WriteLine(filesd);
            //wssv.SslConfiguration.ServerCertificate = new
X509Certificate2(filesd, "Aboba34");
            //wssv.Log.Level = LogLevel.Trace;
            //wssv.SslConfiguration.EnabledSslProtocols = SslProtocols.Tls12;
            //ServicePointManager.ServerCertificateValidationCallback +=
(object sender, X509Certificate cer, X509Chain chain, SslPolicyErrors
sslPolicyErrors) =>
            //{
            //    return true;
            //};
            wssv.AddWebSocketService<Connect>("/connect");
            wssv.AddWebSocketService<Chat>("/chat");
            wssv.AddWebSocketService<Login>("/login");
        }
    }
}

```

```

wssv.Start();
if (wssv.IsListening)
{
    Console.WriteLine("Listening on port {1}:{0}, and providing
WebSocket services:", wssv.Port, wssv.Address);
    foreach (var path in wssv.WebSocketServices.Paths)
        Console.WriteLine("- {0}", path);
}
Console.WriteLine("\nPress Enter key to stop the server...");
Console.ReadLine();
wssv.Stop();
}
}
}
Connect:

```

```

private string GetJSON(Status code, string data, string[] args =
null)
{
    int coded = (int)code;
    var o = JObject.Parse($"{{'code':{coded}}}"");
    o["data"] = JToken.FromObject(data);
    if (args != null)
    {
        if(args.Length % 2 == 0)
        {
            for(int i = 0; i < args.Length - 1; i++)
            {
                o[args[i]] = JToken.FromObject(args[++i]);
            }
        }
    }
    return o.ToString();
}

```

```

protected override void OnMessage(MessageEventArgs e)
{
    dynamic M = JsonConvert.DeserializeObject(e.Data);
    int code = int.Parse($"{M.code}");
    string msg = $"{M.msg}";
    switch (code)
    {
        case -1:
        {
            if (user.connect is Connect)
            {
                user.connect.Send(GetJSON(Status.Error,
"reconnect"));
                user.connect = this;
                goto case 0;
            }
            break;
        }
        case 0://отримати список кімнат
        {
            var userRooms = user.Rooms;

```

```

        r.IsPublic == true);
        var publicRooms = db.GetAllRooms().Where(r =>
//int allRoomsCount = publicRooms.Count();
int allRoomsCount = 473;
int minus = 0;
int page = int.Parse(msg);
if(allRoomsCount > 100)
{
    publicRooms = publicRooms.Skip(page *
100).Take(100);
}
List<dynamic> rooms = new();
foreach (Room r in userRooms.Concat(publicRooms))
{
    dynamic data = new JObject();
    data.name = r.Name;
    data.count = r.connects.Count;
    data.id = r.Id;
    data.isPublic = r.IsPublic;
    data.userRoom = userRooms.Contains(r);
    if ((bool)data.userRoom)
    {
        minus++;
    }
    if (!rooms.Any(room => room.id == r.Id))
    {
        rooms.Add(data);
    }
}
var json = JsonConvert.SerializeObject(rooms);
string message = GetJSON(Status.MyRooms, json,
    new string[]{
        "count", (allRoomsCount - minus).ToString(),
        "page", page.ToString(),
    });
user.connect.Send(message);
break;
}
case 1://створити кімнату
{
    dynamic roomInfo = JsonConvert.DeserializeObject(msg);
    string name = roomInfo.name;
    string pass = roomInfo.pass;
    string url = roomInfo.url;
    bool isPublic = roomInfo.isPublic;
    Room room = db.AddRoom(name, pass, url, user.Hash,
isPublic);

    user.Rooms.Add(room);
    db.SaveChangesAsync();
    dynamic data = new JObject();
    data.name = room.Name;
    data.count = room.connects.Count;
    data.id = room.Id;
    var json = JsonConvert.SerializeObject(data);
    string message = GetJSON(Status.Room, json);
    user.connect.Send(message);
    break;
}

```

```

    }
    case 2://увійти до кімнати
    {
        long id = long.Parse(msg);
        Room room = db.GetRoom(id);
        if(room == null)
        {
            Send(GetJSON(Status.Error, "rooms"));
            return;
        }
        if (!user.Rooms.Contains(room))
        {
            if (string.IsNullOrEmpty(room.Pass))
            {
                user.Rooms.Add(room);
                db.SaveChangesAsync();
            }
            else
            {
                _room = room.Id;
                Send(GetJSON(Status.Pass, "true"));
                return;
            }
        }
        Send(ReturnRoomInfo(room));
        break;
    }
    case 3:
    {
        Room room = user.Rooms.First(x => x.Id ==
long.Parse(msg));
        user.Rooms.Remove(room);
        db.SaveChangesAsync();
        dynamic data = new JObject();
        data.roomId = room.Id;
        data.existence = false;
        Send(GetJSON(Status.ChangeListRooms,
JsonConvert.SerializeObject(data)));
        break;
    }
    case 4:
    {
        Room room = db.GetRoom(_room);
        if(room.Pass == msg)
        {
            user.Rooms.Add(room);
            db.SaveChangesAsync();
            Send(ReturnRoomInfo(room));
        }
        else
        {
            Send(GetJSON(Status.Pass, "false"));
        }
        break;
    }
    case 5:
    {

```

```

dynamic userInfo = JsonConvert.DeserializeObject(msg);
switch (userInfo.var.ToString())
{
    case "name":
    {
        if (string.IsNullOrEmpty(userInfo.name))
        {
            Send(GetJSON(Status.Error, "ErrorData"));
            break;
        }
        user.Name= userInfo.name;
        db.SaveChangesAsync();
        break;
    }
    case "loginWithToken":
    {
        user.LoginWithToken =
bool.Parse(userInfo.value.ToString());
        db.SaveChangesAsync();
        break;
    }
    case "mail":
    {
        user.Email = userInfo.mail;
        db.SaveChangesAsync();
        break;
    }
    case "pass":
    {
        user.Password = userInfo.pass;
        db.SaveChangesAsync();
        break;
    }
    case "reg":
    {
        Regex regex =
new(@"^[^\s@]+@[^\s@]+\.[^\s@]+$");
        if (regex.IsMatch(userInfo.email.ToString()))
        {
            user.Email = userInfo.email;
            user.Password = userInfo.pass;
            db.SaveChangesAsync();
            SendUserInfo();
        }
        else
        {
            Send(GetJSON(Status.Error, "ErrorData"));
            break;
        }
        break;
    }
}
break;
}
}
}

```

Chat.cs:

```
protected override void OnOpen()
{
    var hash = QueryString["hash"];
    var user = db.GetUser(hash);
    if (user == null)
    {
        Sessions.SendTo(GetJSON(Status.Error, "No User"), this.ID);
        return;
    }
    else
    {
        if(user.connect is Connect)
        {
            Connect.Send(user.connect, GetJSON(Status.Chat, ""));
        }
        user.connect = this;
        this.user = user;
    }
    if (!long.TryParse(QueryString["roomId"], out long RoomId))
    {
        Sessions.SendTo(GetJSON(Status.Error, "Wrong code"), this.ID);
        return;
    }
    if (user.Rooms.Find(x=>x.Id == RoomId) == null)
    {
        Sessions.SendTo(GetJSON(Status.Error, "Not your room"),
this.ID);
        return;
    }
    room = db.GetRoom(RoomId);
    if(room == null)
    {
        Sessions.SendTo(GetJSON(Status.Error, "this room does not
exist"), this.ID);
        return;
    }
    dynamic data = new JObject();
    data.code = RoomId;
    data.ctime = room.CurrentTime.ToString().Replace(',', '.');
    data.me = ID;
    admin = db.GetUser(room.Admin).connect.ID;
    data.admin = admin;
    data.pass = room.Pass;
    data.eyesID = ID;
    data.ID = ID;
    data.userList = GetAllUsers(room);
    string mes = GetJSON(Status.Connect, data.ToString());
    Sessions.SendTo(mes, this.ID);
    EnterRoom();
}

protected override void OnMessage(MessageEventArgs e)
{
    dynamic M = JsonConvert.DeserializeObject(e.Data);
    int code = int.Parse($"{M.code}");
```

```

string msg = $"{M.msg}";
switch (code)
{
    case 1:
    {
        if (ID != admin) return;
        dynamic stuff = JsonConvert.DeserializeObject(msg);
        string choice = stuff.action;
        switch (choice)
        {
            case "pause":
            {
                bool value = ($"{stuff.value}" == "-1");
                string id = stuff.id;
                dynamic info = new JObject();
                info.name = "all";
                info.value = value;
                if (id == ID)
                {
                    room.canPlay = value;
                    SentToGroop(Status.Info,
Lang.GetValue("CanPause", info));
                }
                else
                {
                    Chat user = room.connects.Find(x => x.ID
== id);
                    user.canPause = value;
                    SentToGroop(Status.Info,
Lang.GetValue("CanPause", info));
                }
                break;
            }
            case "kick":
            {
                string id = stuff.id;
                Chat user = room.connects.Find(x => x.ID ==
id);
                user.Send(GetJSON(Status.Page, "kick"));
                break;
            }
        }
        break;
    }
    case 3:
    {
        SentToGroop(Status.Chat, msg, new string[] { "name",
user.Name });
        break;
    }
    case 4:
    {
        dynamic stuff = JsonConvert.DeserializeObject(msg);
        bool value = stuff.value;
        string type = stuff.type;

```

```

        double time = stuff.time;
        if (!canPause)
        {
            Send(GetJSON(Status.Error,
Lang.GetValue("CantPlay"))));
            return;
        }
        switch (type)
        {
            case "status":
            {
                dynamic data = new JObject();
                data.value = value;
                data.id = ID;
                SentToGroop(Status.StatusPlayer,
data.ToString());

                break;
            }
            case "play":
            {
                if (!value)
                {
                    SentToGroop(Status.Player, msg);
                    room.UpdateTimer(time, false);
                }
                else
                {
                    SentToGroop(Status.ReadyPlay, msg);
                    Console.WriteLine("Проверка готовности");
                }
                break;
            }
            case "seeked":
            {
                if (M.duration != room.Duration)
                {
                    room.UpdateTimer(time, false);
                    SentToGroop(Status.Player, msg);
                }
                break;
            }
            case "rate":
            {
                SentToGroop(Status.Player, msg);
                break;
            }
        }

        break;
    }
    case 9:
    {
        ready = true;
        var check = room.connects.Find(x => x.ready == false);
        if (check == null)

```

```

        {
            dynamic stuff = JsonConvert.DeserializeObject(msg);
            double time = stuff.time;
            room.UpdateTimer(time, true);
            Console.WriteLine("VKL");
            SentToGroop(Status.Player, msg);
            foreach (var a in room.connects)
            {
                a.ready = false;
            }
        }
        break;
    }
}

private void SentToGroop(Status code, string msg, string[] args =
null)
{
    string res = GetJSON(code, msg, args);
    foreach (Chat id in room.connects)
    {
        Sessions.SendTo(res, id.ID);
    }
}

```

DataBase.cs:

```

protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
{
    optionsBuilder.UseNpgsql("Host=localhost;Port=5432;Database=4eyes;Username
=postgres;Password=123");
}

public static DataBase ConnectToDB()
{
    if (db == null)
    {
        db = new DataBase();
        var b12 = db.Rooms.Include(c => c.Users).ToList();
        return db;
    }
    else
    {
        return db;
    }
}

public void AddUser(User user)
{
    User newUser = GetUser(user.Hash);
    if (newUser == null)
    {
        db.Users.Add(user);
        db.SaveChanges();
    }
}

```

```

    }
    db.SaveChanges();
}
public User GetUser(string hash)
{
    return db.Users.FirstOrDefault(x => x.Hash == hash);
}
public User GetUserbyLogin(string login)
{
    return db.Users.FirstOrDefault(x => x.Email == login);
}
public List<User> GetAllUsers()
{
    return db.Users.ToList();
}
public Room AddRoom(string Name, string Pass, string Url, string
Admin, bool isPublic)
{
    long id;
    Random rnd = new();
    do
    {
        id = rnd.Next(100000, 999999);

        } while (db.GetRoom(id) != null);
    Room room = new Room(id, Name, Pass, Url, Admin, isPublic);
    db.Rooms.Add(room);
    db.SaveChanges();
    return room;
}
public Room GetRoom(long id)
{
    return db.Rooms.FirstOrDefault(x => x.Id == id);
}
public List<Room> GetAllRooms()
{
    return db.Rooms.ToList();
}
public int CountUsersInRoom(long id)
{
    Room room = GetRoom(id);
    if(room == null)
    {
        return -1;
    }
    else
    {
        return room.Users.Count;
    }
}
}

```