МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
Т.в.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 «Комп'ютерні науки»_____,

освітньо-наукової програми «Інформаційні технології проектування»_____

на тему: *Програмний додаток створення макету приміщення для моделювання інтер'єру та комунікацій "VR/ARoom". Хмарний додаток демонстрації та редагування дизайну приміщення з використанням VR технологій*

Здобувача групи ІТ.м-11н Толстоноженка Станіслава Олеговича_____
_____
(шифр групи) (прізвище, ім'я, по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Станіслав ТОЛСТОНОЖЕНКО___
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник _____к.т.н., доц. Юлія ПАРФЕНЕНКО _____ _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-наукова програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ

В.о. зав. кафедри ІТ

_ С. М. Ващенко

«___»_____2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Толстоноженко Станіслав Олегович
(прізвище, ім'я, по батькові)

1 Тема проекту *Програмний додаток створення макету приміщення для моделювання інтер'єру та комунікацій "VR/ARoom". Хмарний додаток демонстрації та редагування дизайну приміщення з використанням VR технологій*

затверджена наказом по університету від « 05 » травня 2023 р. № 0465-VI

2 Термін здачі студентом закінченого проекту «_12_» ___травня___ 2023 р.
3 Вхідні дані до проекту *технічне завдання на розробку хмарного додатку демонстрації та редагування дизайну приміщення з використанням VR технологій*

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *аналіз предметної області, моделювання, проектування програмного додатку, реалізація хмарного додатку демонстрації та редагування дизайну приміщення з використанням VR технологій*

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

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	**Консультант**	**Підпис, дата**	
		Завдання видав	**Завдання прийняв**

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання ___
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускної проекту	Термін виконання етапів проекту	Примітка
1	Розробка концепції проекту	18.01.23-20.01.23	
2	Обговорення роботи з керівником	21.01.23-23.01.23	
3	Затвердження теми	24.01.23-25.01.23	
4	Виконання розділу «Керування проектами»	26.01.23-08.02.23	
5	Розробка додатку	09.02.23-15.04.23	
6	Тестування додатку	16.04.23-21.04.23	
7	Розробка документації	15.02.23-12.05.23	

Магістрант _____ Толстоноженко С.О.

Керівник роботи _____ к.т.н., доц. Парфененко Ю.В.

РЕФЕРАТ

Тема кваліфікаційної роботи магістра «Програмний додаток створення макету приміщення для моделювання інтер'єру та комунікацій "VR/ARoom". Хмарний додаток демонстрації та редагування дизайну приміщення з використанням VR технологій».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 34 найменувань, додатку. Загальний обсяг роботи − 65 сторінок, у тому числі 47 сторінки основного тексту, 4 сторінки списку використаних джерел, 8 сторінок додатку.

Кваліфікаційну роботу магістра присвячено розробці програмного забезпечення для обміну даними між додатками, що працюють на базі різних операційних систем.

В роботі проведено аналіз предметної області, оглянуто останні дослідження та публікації, проаналізовано аналогічні додатки і визначено їх можливості і недоліки.

У роботі виконано аналіз актуальності досліджуваної задачі, проаналізована робота програмних продуктів-аналогів, розроблена мета та опрацьовані задачі проекту, а також здійснено проектування програмного продукту з використанням UML.

Результатом проведеної роботи є розроблений програмний додаток для обміну даними між додатками, що працюють на базі різних операційних систем.

Практичне значення роботи полягає у тому, що використання розробленого додатку дозволить створювати підключення через веб-сокет та надавати необхідну інформацію для P2P підключення, щоб мати можливість передавати дані між додатками як на базі однієї платформи так і на різних.

Ключові слова: C++, Unreal Engine, JavaScript, NodeJS, PeerToPeer, сервер, клієнт, віртуальна реальність.

ЗМІСТ

ВСТУП 6
1 ОГЛЯД ЛІТЕРАТУРИ 8
1.1 Актуальність досліджуваної задачі 8
1.2 Аналіз програмних продуктів-аналогів 13
2 ПОСТАНОВКА ЗАДАЧІ ТА ОПИС МЕТОДІВ ДОСЛІДЖЕННЯ 17
2.1 Мета та задачі дослідження 17
3 ПРОЕКТУВАННЯ ПРОГРАМНОГО ДОДАТКУ 19
3.1 Моделювання програмного додатка в нотації IDEF0 19
3.2 Модель аналізу предметної області програмного додатку 23
3.3 Модель проектування програмного додатку 26
3.5 Модель реалізації 29
3.6 Моделювання даних 30
4 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ДОДАТКУ 33
4.1 Розробка програмного продукту 33
4.2 Дослідження характеристик передачі даних 45
ВИСНОВКИ 51
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 53
Додаток А 57

ВСТУП

Актуальність. Передача та обмін даними відіграють вирішальну роль у сучасному цифровому світі, де інформація є цінною одиницею [1]. Здатність безперебійно передавати та обмінюватися даними між системами та пристроями стала важливою для ефективної та результативної роботи компаній та окремих осіб. А віртуальна реальність (VR) надає імерсійний досвід, що дозволяє користувачам зануритися в цифрове середовище [2]. Вона застосовується у багатьох галузях, включаючи ігрову індустрію, навчання, медицину, архітектуру, туризм та інші [3]. Об’єднання цих двох технологій може надати можливість створити спільний сервер, до якого будуть підключатися користувачі через додатки з підтримкою віртуальної реальності для перегляду та взаємодії з тривимірними об’єктами. В даному випадку буде розглядатися питання сумісності різних систем у процесі обміну даними. А саме можливі прямого обміну даними між додатками на базі різних платформ, даному випадку розглядається веб-сервер та VR додаток для демонстрації роботи підключення. Пряме підключення між додатками дозволяє уникати використання серверів посередників. При використанні такого підходу спрощується архітектура інформаційної системи, зменшується затримка з’єднання між клієнтами мережі.

Об’єкт дослідження. Процес обміну даними між користувачами з використанням прямого підключення.

Предмет дослідження. Методи, технології та інструменти для розробки хмарного додатку та плагіну, спрямовані на забезпечення взаємодії між додатками, які працюють на однакових та різних операційних системах.

Наукова новизна. На відміну від існуючих методів передачі даних, які використовують технологію прямого підключення, розроблений метод дозволить

інтегрувати функцію файлообміну у додатки шляхом підключення його у вигляді плагіну.

Гіпотеза дослідження. Якщо розробити універсальний метод обміну файлами шляхом прямого підключення, можна буде обмінюватися даними між додатками або сервісами на базі різних платформ без використання додаткових серверів, які обробляють проміжні дані, тим самим зменшуючи залежність від сторонніх постачальників послуг і потенційні ризики для конфіденційності даних користувача.

Мета. Розробити програмне забезпечення для обміну даними між додатками, що працюють на базі різних операційних систем.

Основні задачі. Основними задачами кваліфікаційної роботи магістра є:

- Дослідити предметну область;
- Сформулювати постановку задачі;
- Моделювати та проектувати розроблюваний додаток;
- Реалізувати додаток програмно;
- Дослідити основні характеристики процесу передачі даних між клієнтами з використанням веб-серверу та додатку з підтримкою віртуальної реальності на базі Unreal Engine для платформи Windows.;
- Сформулювати висновки.

Практичне значення. Використання розробленого додатку дозволить створювати підключення через веб-сокет та надавати необхідну інформацію для P2P (Peer-to-Peer, що на українською означає однорангова мережа) підключення, щоб мати можливість передавати дані між додатками як на базі однієї платформи так і на різних.

Апробація результатів. Доповідь результатів проводилася на науково технічній конференції «Інформатика Математика Автоматика», м. Суми: СумДУ, 27-28 квітня 2023.

1 ОГЛЯД ЛІТЕРАТУРИ

1.1 Актуальність досліджуваної задачі

Технологія передачі даних все ще залишається дуже актуальною та важливою сферою технологій сьогодні і її значення з часом лише зростає. А її використання у поєднанні з технологією віртуальної реальності дозволить створювати віртуальні кімнати для відвідування користувачами, які мають доступ до мережі інтернет та необхідне обладнання [1].

Обсяг даних, які генеруються та обмінюються в усьому світі, зростає високим темпом, а компанії та окремі особи все більше покладаються на безперебійну та безпечну передачу даних для ефективної роботи [4].

Крім того, прогрес у технологіях уможливив більш швидку та ефективну передачу даних і методи обміну, що забезпечують, швидку та безпечну передачу великих обсягів даних [5]. Наприклад, високошвидкісне підключення до Інтернету, хмарне зберігання та обчислення значно розширили можливості передачі та обміну даними [6].

Але не зважаючи на поточні досягнення у технологій передачі та обміну даними є певні недоліки, а саме:

- питання безпеки та конфіденційності інформації, її захист від зовнішнього втручання;
- проблеми сумісності форматів файлів, протоколів або стандартів передачі та обміну даними;
- обмеження пропускної здатності мережі під час передачі файлів, особливо дане явище критично впливає на файли великого об’єму, адже затримка передачі даних може вплинути на ефективність роботи поточної системи вцілому;

– цілісність даних може пошкоджуватися у процесі передачі, тому її забезпечення є одним із критичних етапів розробки інформаційної системи.

Також варто зауважити, що пандемія COVID-19 ще більше показала важливість технології передачі даних. Оскільки віддалена робота та онлайн зустрічі все більше використовуються в повсякденному житті, передача та обмін даними стали ще важливішими для компаній і окремих осіб для ефективної роботи та безперебійного спілкування [7].

Серед наукових статей дана тема також є досить актуальною, адже за даними платформи Mendeley кожного року публікується близько 50 досліджень [8].

Наприклад розглянемо публікацію «Providing security and fault tolerance in P2P connections between clouds for mHealth services» [9]. Її завданням було дослідження безпеки обміну даними та надійність підключення між хмарними сервісами mHealth. Результати показують, що завдяки особливості технології прямого підключення стабільність підключення є високою, адже з’єднання відбуваються безпосередньо між сервісами, без додаткових посередників, що в свою чергу позитивно впливає на безпеку обміну даними.

У публікації «Efficient neighbor selection through connection switching for P2P live streaming» сконцентрована увага на можливостях використання прямого підключення для потокових трансляцій [10]. До цього поняття можна віднести такі дії, як трансляція відео, звуку, обмін повідомленнями. Також звертається увага на високу масштабованість та невисокі вимоги для реалізації даної технології, адже у ролі серверів виступає кожен з активних користувачів.

У публікації «Exploring WebRTC Potential for DICOM File Sharing» було досліджено переваги використання прямого підключення для передачі даних між складовими частинами сервісу DICOM. Увага спрямована на те, що процес реалізації прямого підключення проводиться без додаткових витрат на забезпечення відокремленого серверу [11].

У публікації «An adaptive bitrate switching algorithm for speech applications in context of webrtc» досліджується можливість покращення стабільності роботи потокових трансляцій. Зокрема, було розглянуто метод адаптивного налаштування бітрейту, а саме реалізація методу обчислення оптимальної якості відео та звуку в залежності від пропускної здатності інтернет мережі [12].

У публікації «Cost-effective load testing of WebRTC applications» дослідліджено шляхи проведення тестування навантаження на додатки з прямим підключенням. Було розглянуто два варіанти, а саме розглянуто доступні методи тестування та їх порівняння до нового розробленого методу [13].

У публікації «Understanding and estimating quality of experience in WebRTC applications» розглядається індекс задоволеності користувачів при користуванні додатками аудіо та відео зв’язку на базі технології прямого підключення [14].

У публікації «Audio and Video Mixing Method to Enhance WebRTC» показано неможливість використання поточних методів змішування аудіо та відео для їх синхронної трансляції з кількох пірів та подано метод, що дозволяє реалізувати використання технології прямого підключення у такому сценарії [15].

У публікації «WebRTC-based video conferencing service for telehealth» показано недоліки наявних систем для медичних сервісів відеоконференцій. А саме: досить висока ціна придбання та обслуговування, використання у них технологій, що не мають сумісності, що в свою чергу потребує досить досвідчених IT робітників для підтримання роботи системи. Тому було розглянуто можливість застосування технології прямого підключення на базі JavaSctipt API для даного варіанту використання, адже вона дозволяє значно спростити архітектуру додатку [16].

У публікації «A Comparison of WebRTC and Conventional Videoconferencing for Synchronized Remote Medical Image Presentation» проведено порівняння методів віддаленого показу зображень для медичних цілей при використанні додатку для

відеоконференцій на основі технології прямого підключення та звичайної архітектури з використанням віддаленого серверу [17].

У публікації «Improved Jitter Buffer Management for WebRTC» досліджено можливість створення покращеного методу буферизації для більш стабільної передачі відео та аудіо в додатку для відеоконференцій. Даний засіб дозволяє уникати переривання передачі аудіо та відео при короткочасних погіршеннях підключення до мережі Інтернет [18].

У публікації «WebRTC security measures and weaknesses» досліджено розроблені на поточний час механізми захисту даних та їх недоліки при передачі даних через технологію прямого підключення [19].

У публікації «Recognition of traffic generated by WebRTC communication» досліджено можливість проведення розпізнавання трафіку від технології прямого з'єднання для можливості операторів мережі надавати йому пріоритет, відслідковувати безпеку з'єднання, визначати проблеми, що виникають [20].

В результаті огляду предметної області даної роботи та наукових публікацій щодо обраної теми, було визначено, що в переважній більшості додатків та досліджень показано використання технології прямого підключення в якості засобу обміну даними між користувачами відеоконференцій та її використання лише межах певної інформаційної системи. В той же час, серед опрацьованого матеріалу не було розглянуто можливісті її використання у вигляді мультиплатформного інтегрованого модуля для одночасного використання веб-сервісів та програмних додатків (рис. 1.1).

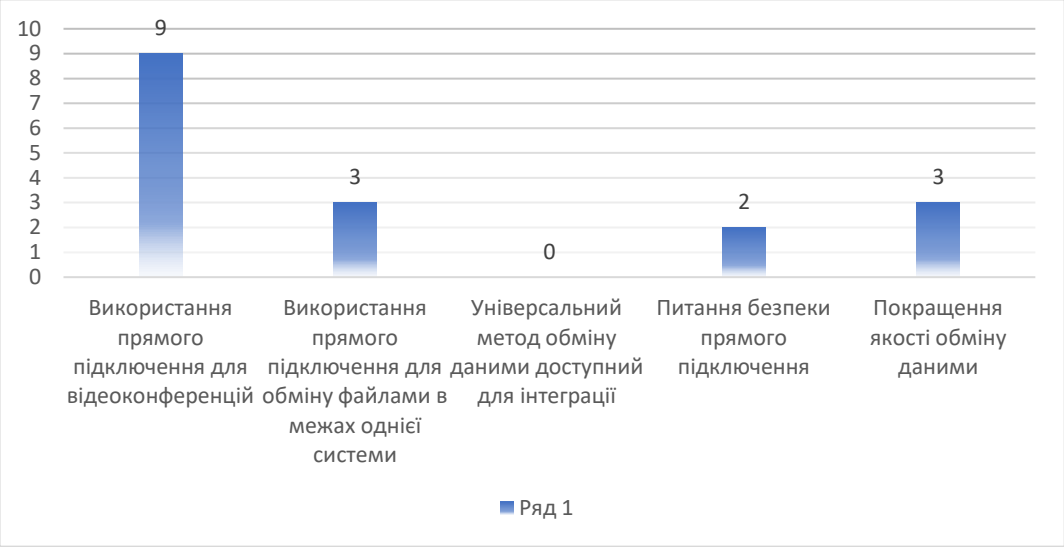

Рисунок 1.1 Перелік розглянутих питань у наукових публікаціях

Отже, виникає необхідність створення програмного засобу, використання якого дозволить забезпечити обмін даними між додатками на базі різних платформ, будь то веб-сервіс чи додаток на базі Windows. Реалізація даної можливості дозволить покращити певні аспекти передачі та обміну даними, а саме:

- Пряме підключення між клієнтами;
- Підтримка передачі даними між різними платформами;
- Забезпечення безпеки та конфіденційності при передачі даних;
- Оптимізація швидкості передачі та зменшення втрат даних;
- Можливість синхронізації даних між пристроями та автоматичне резервне копіювання;

1.2 Аналіз програмних продуктів-аналогів

У процесі аналізу програмних продуктів-аналогів, які мають можливість прямого підключення, були досліджені такі додатки, як WebTorrent, Dat Project, ShareDrop.

WebTorrent — це проект із відкритим кодом, який забезпечує одноранговий (P2P) обмін файлами безпосередньо у веб-браузері за допомогою протоколу BitTorrent [21]. Це дозволяє користувачам обмінюватися файлами та передавати їх іншим користувачам без необхідності використання центрального сервера. WebTorrent використовує WebRTC (Web Real-Time Communication) для встановлення прямих з'єднань між одноранговими користувачами та обміну даними, що дає змогу передавати великі файли ефективніше, ніж традиційні методи клієнт-сервер. WebTorrent доступний як бібліотека JavaScript, яку можна використовувати у веб-додатках, що дозволяє розробникам створювати функції обміну файлами P2P (peer to peer, тобто пряме підключення) безпосередньо на своїх веб-сайтах або у веб-додатках. Він також може підтримувати потокове передавання аудіо- та відеовмісту в реальному часі, що робить його придатним варіантом для медіа-додатків.

Схему роботи даного додатку показано на рис. 1.2.

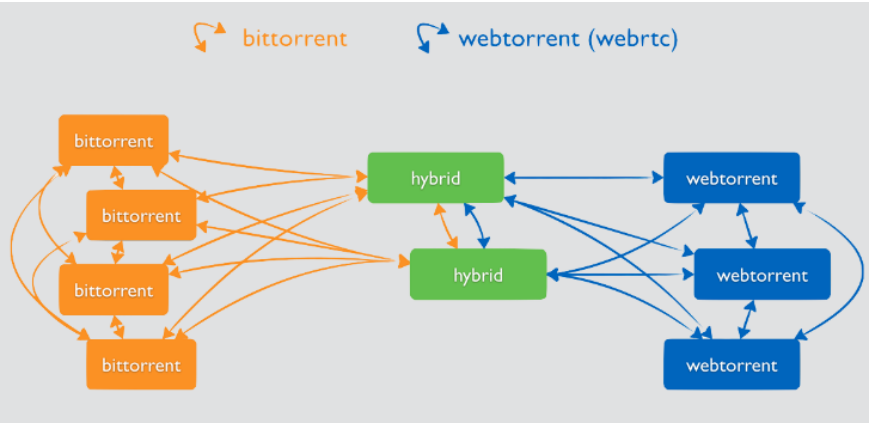

Рисунок 1.2 – Додаток WebTorrent

Dat Project — це децентралізована платформа обміну даними з відкритим кодом, яка дозволяє користувачам безпечно та анонімно ділитися файлами та синхронізувати їх [22]. Dat Project використовує протокол Dat, який є одноранговим протоколом передачі даних, розробленим для розподілених програм. Протокол Dat використовує розподілену хеш-таблицю (DHT) для полегшення виявлення та передачі файлів між одноранговими вузлами без необхідності центрального сервера. Платформа забезпечує наскрізне шифрування та забезпечує безпеку та конфіденційність даних.

Dat Project також підтримує одночасну роботу декількох користувачів в реальному часі та контроль версій, що робить його придатним для спільної роботи. Він доступний як інструмент командного рядка та бібліотека JavaScript, що дозволяє розробникам створювати децентралізовані програми, які можуть працювати з протоколом Dat. Крім того, Dat Project має відкритий вихідний код і вільний для використання, що робить його доступним для всіх, хто хоче ділитися та передавати дані безпечним і децентралізованим способом.

ShareDrop — це веб-програма P2P для обміну файлами з відкритим кодом, яка дозволяє користувачам передавати файли безпосередньо між пристроями, підключеними до однієї мережі, без підключення до Інтернету [23]. ShareDrop використовує технологію WebRTC для встановлення прямого з’єднання між пристроями та дозволяє користувачам перетягувати файли на веб-сторінку для їх передачі.

ShareDrop є повністю веб-інтерфейсом і не потребує встановлення чи реєстрації, що робить його простим і зручним способом безпечної та швидкої передачі файлів. Платформа доступна безкоштовно і може використовуватися на будь-якому пристрої з веб-браузером. ShareDrop також має відкритий вихідний код, а це означає, що його код доступний для будь-кого, щоб перевірити, змінити чи зробити свій внесок.

Загалом ShareDrop — це швидке, просте у використанні та безкоштовне рішення для обміну файлами P2P, яке ідеально підходить для користувачів, яким потрібно передавати файли між пристроями в одній мережі без використання Інтернету.

Більш детальне порівняння показане у таблиці 1.1.

Таблиця 1.1 Порівняння продуктів аналогів

Критерій	WebTorrent	Dat Project	ShareDrop
Платформа	Робиться підтримка веб-браузерів та Node.js	Підтримується на різних платформах, Windows, macOS та Linux	Робиться підтримка веб-браузерів
Функції	Надає можливість додавання та відстеження торрент-файлів	Дозволяє розповсюджувати файли з можливістю синхронізації та версіонування	Дозволяє відправляти файли безпосередньо в браузері
Захист даних	Використовується шифрування даних та обмін ключами Diffie-Hellman	Забезпечується безпека за допомогою енкрипції та підписування даних	Дані передаються захищеним каналом, використовуючи протокол HTTPS
Відкритість	Відкритий вихідний код та ліцензування MIT	Відкритий вихідний код та ліцензування MIT	Відкритий вихідний код та ліцензування MIT

Дослідження додатків аналогів показало, всі вони використовують технологію прямого підключення WebRTC. Кожен з них надає можливості

пов’язані з обміном даних, проте вони можуть бути реалізовані тільки з використанням певних платформ або засобів. Також до переваг можна віднести наявність протоколів захисту даних та відкритий код додатків для можливості легально інтегрувати їх у власні проект.

Після дослідження продуктів аналогів були визначені недоліки кожного з них.

WebTorrent має обмежену підтримку, оскільки працює лише з деякими типами додатків, які можуть використовувати технологію BitTorrent протоколів. Також відсутня приватність, через надання доступу до публічних файлообмінних мереж , які можуть бути вразливі до стеження та перехоплення даних.

Dat Project має обмежену сферу застосування, оскільки зосереджений на використанні в розподіленому веб-хостингу та контролі версій даних.

ShareDrop працює лише у вигляді самостійного сервісу і лише у локальній мережі без можливості зміни конфігурації.

Після аналізу переваг та недоліків аналогів, було визначено їх сильні та слабкі сторони у використанні прямого підключення. На їх основі був сформований список задач для розробки програмного забезпечення, а саме:

- реалізація з'єднання з вузлами мережі, обмін даними між вузлами та збереження даних;
- реалізація механізмів шифрування даних та автентифікації користувачів для забезпечення безпеки та конфіденційності даних;
- розробка інтерфейсу, що дозволяє користувачам підключатися до мережі та взаємодіяти з нею, зберігати та обмінюватися даними;
- розробка web-сторінки для демонстрації роботи частин програмного продукту.

Фінальним результатом буде програмний продукт, що дозволить уникнути недоліків, які є в наявних аналогах та забезпечить більш просту інтеграцію технології P2P у програмні додатки.

2 ПОСТАНОВКА ЗАДАЧІ ТА ОПИС МЕТОДІВ ДОСЛІДЖЕННЯ

2.1 Мета та задачі дослідження

Метою кваліфікаційної роботи магістра є розробка програмного продукту у вигляді хмарного додатку, який допоможе налагодити зв’язок між додатками, що працюють як на однаковій, так і на різних операційних системах з використанням плагінів і продемонструвати його роботу на прикладі VR додатку на базі операційної системи Windows.

Програма дозволить створювати підключення через веб-сокет та надавати необхідну інформацію для P2P підключення. Ці види підключення є універсальними для різних операційних систем, дозволяють налаштовувати та підтримувати велику кількість підключень, завдяки інтуїтивному інтерфейсу розробники зможуть легко розібратися з роботою плагіна.

Для досягнення мети були сформовані наступні кроки:

- створити сервер за допомого фреймворка Node.js, на якому буде реалізована взаємодія з веб-сокетами;
- створити базу даних для серверу для зберігання всієї необхідної інформації;
- створити веб-сторінки для серверу з ціллю відображення необхідних даних стосовно підключення клієнтів;
- створити плагін для Unreal Engine 4 для демонстрації роботи розробленої технології;
- розмістити проект на хостингу;
- дослідити основні характеристики процесу передачі даних між клієнтами.

Використання розробленого додатку дозволить створювати підключення через веб-сокет та надавати необхідну інформацію для P2P підключення, щоб мати можливість передавати дані між додатками як на базі однієї платформи так і на різних.

Дослідження основних характеристик процесу передачі даних між клієнтами буде проводитися з використанням веб-серверу та додатку з підтримкою віртуальної реальності на базі Unreal Engine для платформи Windows. Будуть проведені наступні сценарії тестування:

- передача файлів між клієнтом та сервером;
- передача файлів між клієнтами з використанням прямого підключення
- перевірка максимальної кількості одночасного підключення клієнтів до серверу.

3 ПРОЕКТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

3.1 Моделювання програмного додатка в нотації IDEF0

3.1.1 Функціональне моделювання програмного додатка в IDEF0

Для функціонального моделювання плагіну, що використовує P2P підключення до хмарного додатку, можна використовувати такі діаграми як:

– контекстна діаграма;

– діаграма верхнього рівня;

– набір дочірніх діаграм, що деталізують окремі функції системи.

Вхідними даними є дані віртуальної кімнати, які будуть містити інформацію для підключення до поточної кімнати.

Обмеження складаються з:

– статичних даних, що містять інформацію про віртуальну кімнату;

– алгоритму обміну даними між користувачами та сервером;

– алгоритму побудови віртуальної кімнати.

Ресурси плагіну формуються з двох серверів, бази даних та програмного додатку.

Результатом роботи буде відредагована тривимірна модель кімнати.

Контекстна діаграма процесу передачі даних у додатку зображено на рис. 3.1 [24].

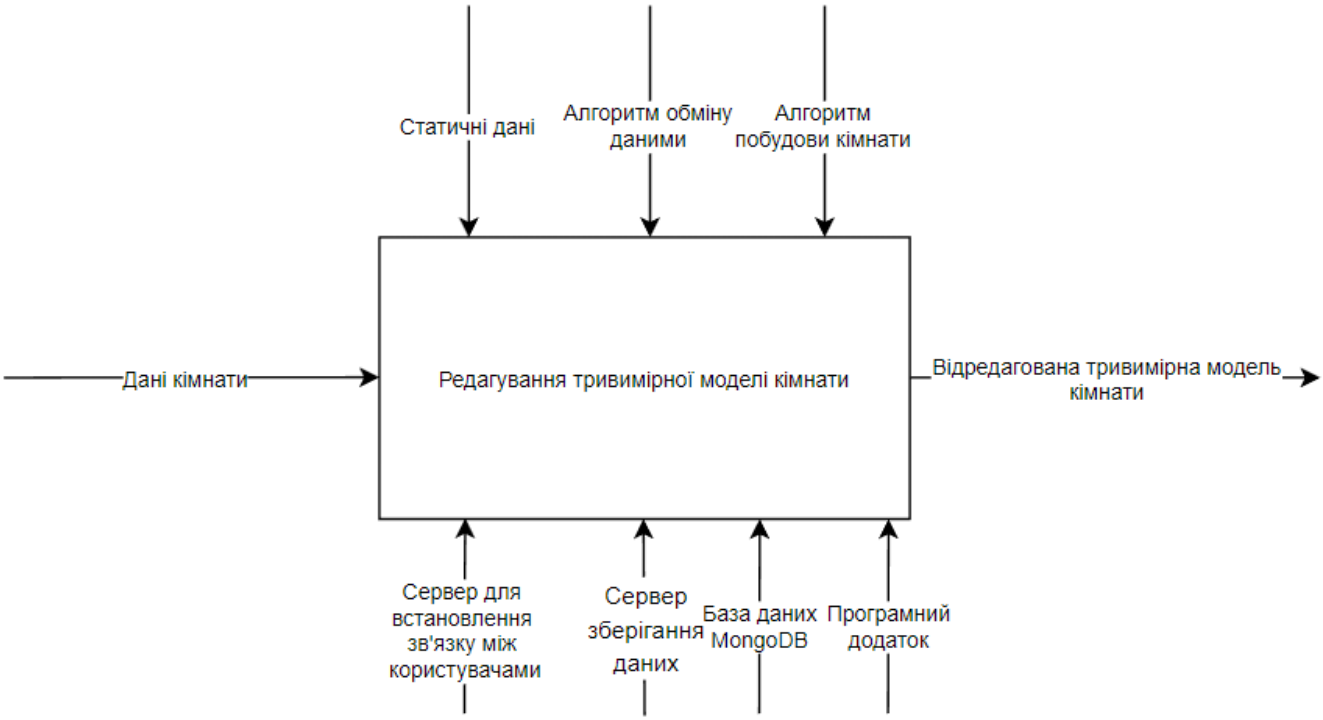

Рисунок 3.1 – Контекстна діаграма процесу редагування моделі кімнати

3.1.2 Декомпозиція першого рівня контекстної діаграми в нотації IDEF0

Декомпозиція першого рівня контекстної діаграми в нотації IDEF0 дозволяє вивчити та проаналізувати взаємозв’язки організацій, змісту та структури [25].

На діаграмі на рис. 3.2 можна побачити покроковий опис роботи додатку.

Спочатку відображається головне меню, що надає можливість перейти до вікна введення даних для підключення до серверу. Після успішного підключення відображається список доступних завантажених віртуальних кімнат. Після входу у віртуальну кімнату відбувається взаємодія користувача з тривимірною моделлю, а внесені зміни зберігаються та стають видимі іншим користувачам.

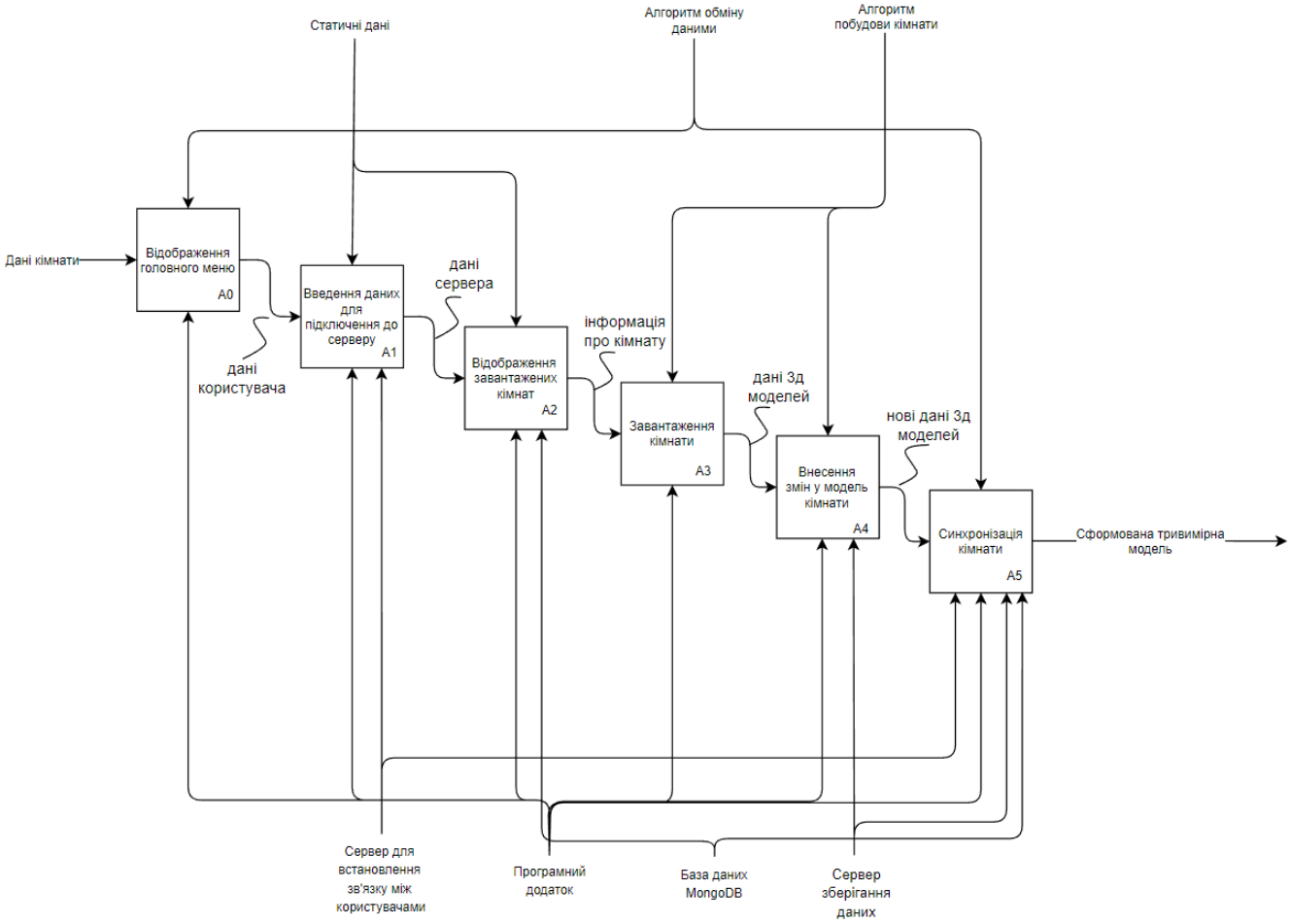

Рисунок 3.2 – Діаграма декомпозиції процесу обміну даними у додатку

3.1.3. Структурне моделювання програмного додатку

Діаграма станів автомата використовується для того, щоб відобразити поведінку об’єкта моделювання протягом його життєвого циклу.

Спочатку для плагіну відбувається підключення до веб-хостингу за допомогою веб-сокетів. Перевірка наявності підключень інших пристроїв. Потім відбувається P2P з'єднання з іншими пристроями для передачі чи отримання даних.

Діаграма автоматів програмного додатку зображено на рис. 3.3.

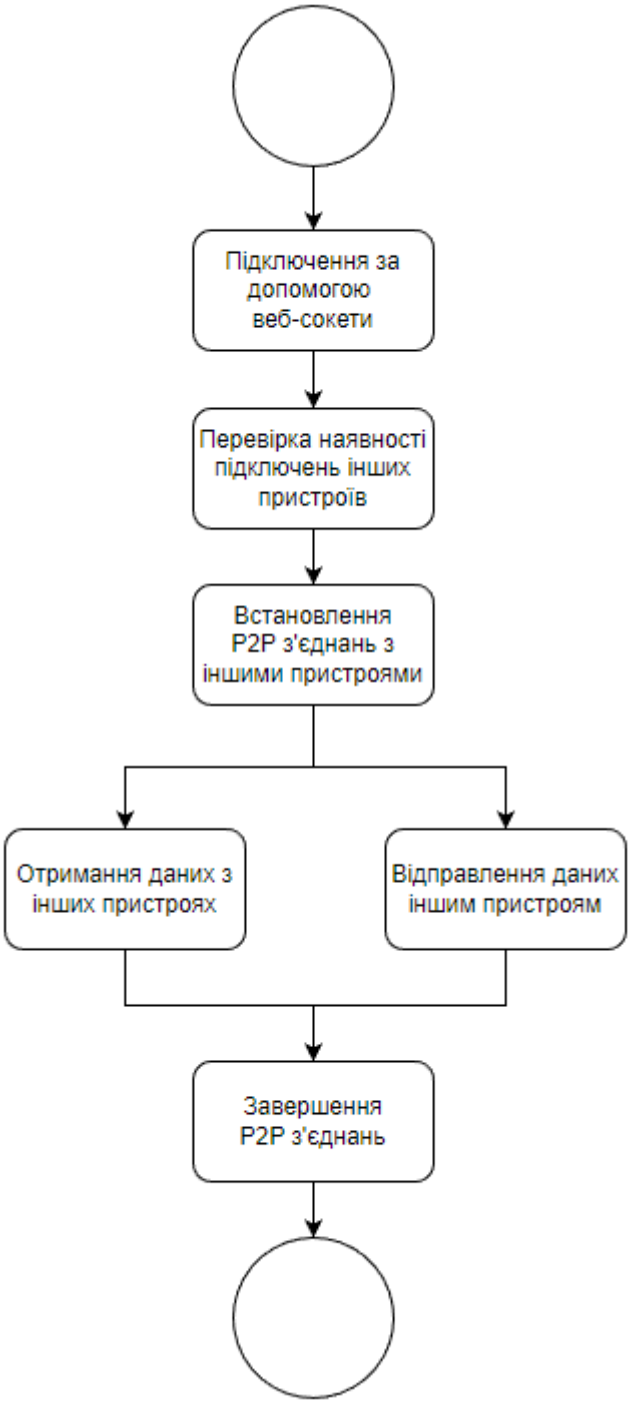

Рисунок 3.3 – Діаграма станів автомата

3.2 Модель аналізу предметної області програмного додатку

3.2.1 Діаграма класів аналізу програмного додатку

У ході моделювання об’єктно-орієнтованих систем застосовується побудова діаграми класів аналізу. Вона відображає сукупність елементів моделі, які відображають операції та відношення, що їх з’єднують [26].

Веб-сервер має головне меню у вигляді веб-сторінки. Має можливість оновлювати дані шляхом підключення до бази даних MongoDB. Надає можливість клієнтам з’єднуватися за допомогою веб-сокетів та обмінюватися даними через нього.

Діаграма класів аналізу програмного додатку зображено на рис. 3.4.

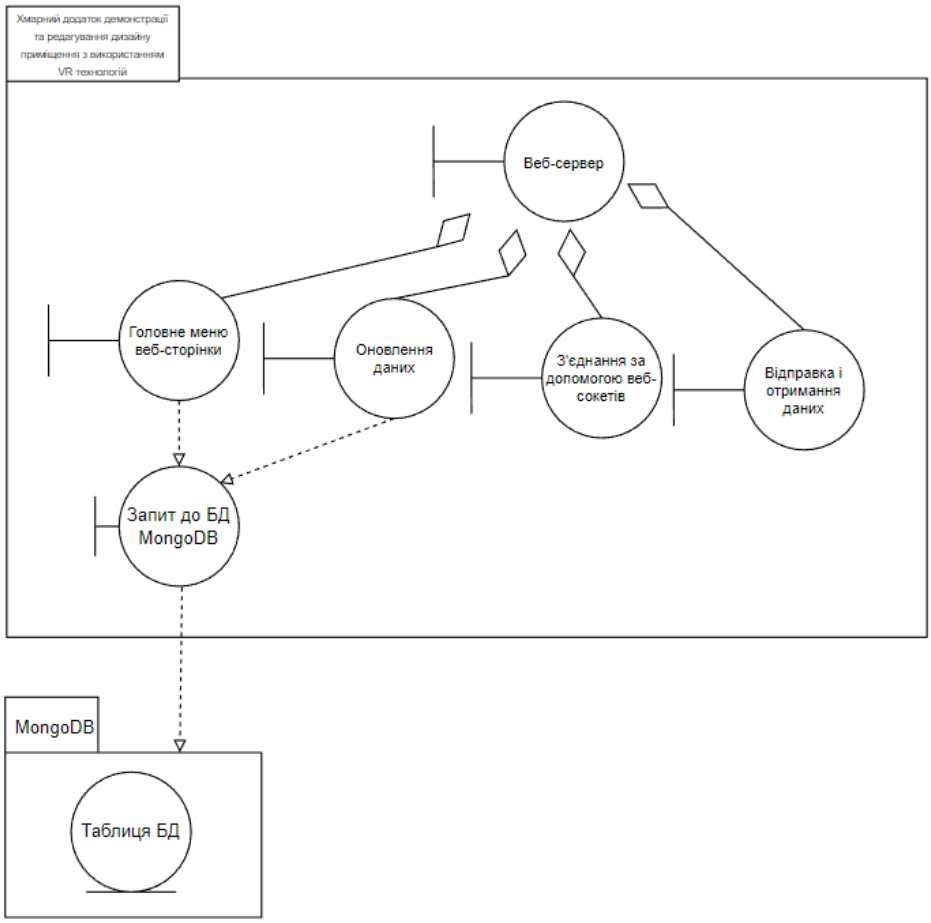

Рисунок 3.4 – Діаграма класів аналізу програмного додатку

3.2.2 Діаграми послідовності.

Діаграми послідовності показують взаємодію об’єктів моделювання відносно часу. Прямокутник зображає об’єкт моделювання та розміщується зверху пунктирної лінії, яка грає роль лінії життя моделі об’єкту [27].

Для забезпечення прямого підключення між клієнтами для майбутньої передачі даних без використання віддалених серверів, спочатку необхідно провести підключення до хмарного додатку, який надає інформацію про користувача, до якого буде відбуватися підключення.

Діаграма послідовності підключення між клієнтом та сервером показана на рисунку 3.5.

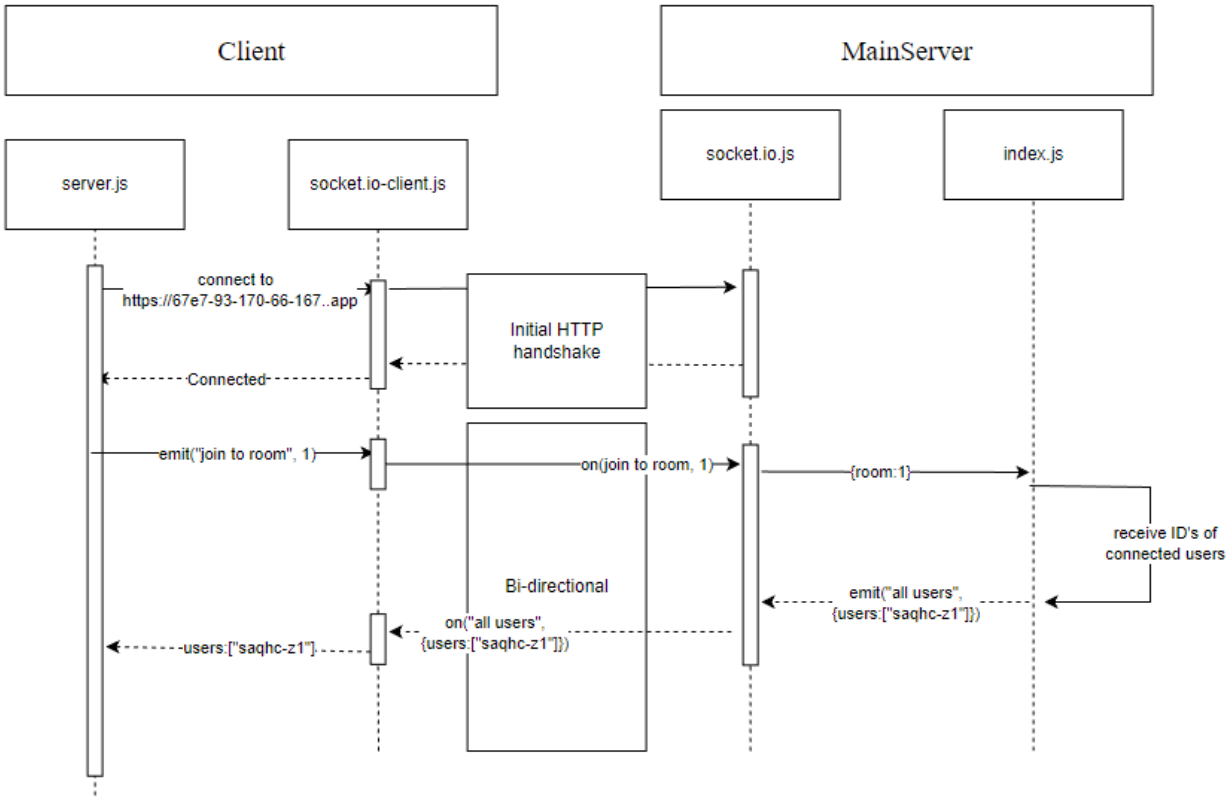

Рисунок 3.5 – Діаграма послідовності підключення між клієнтом та сервером

Пряме підключення та обмін даними між клієнтами показане на рисунку 3.6.

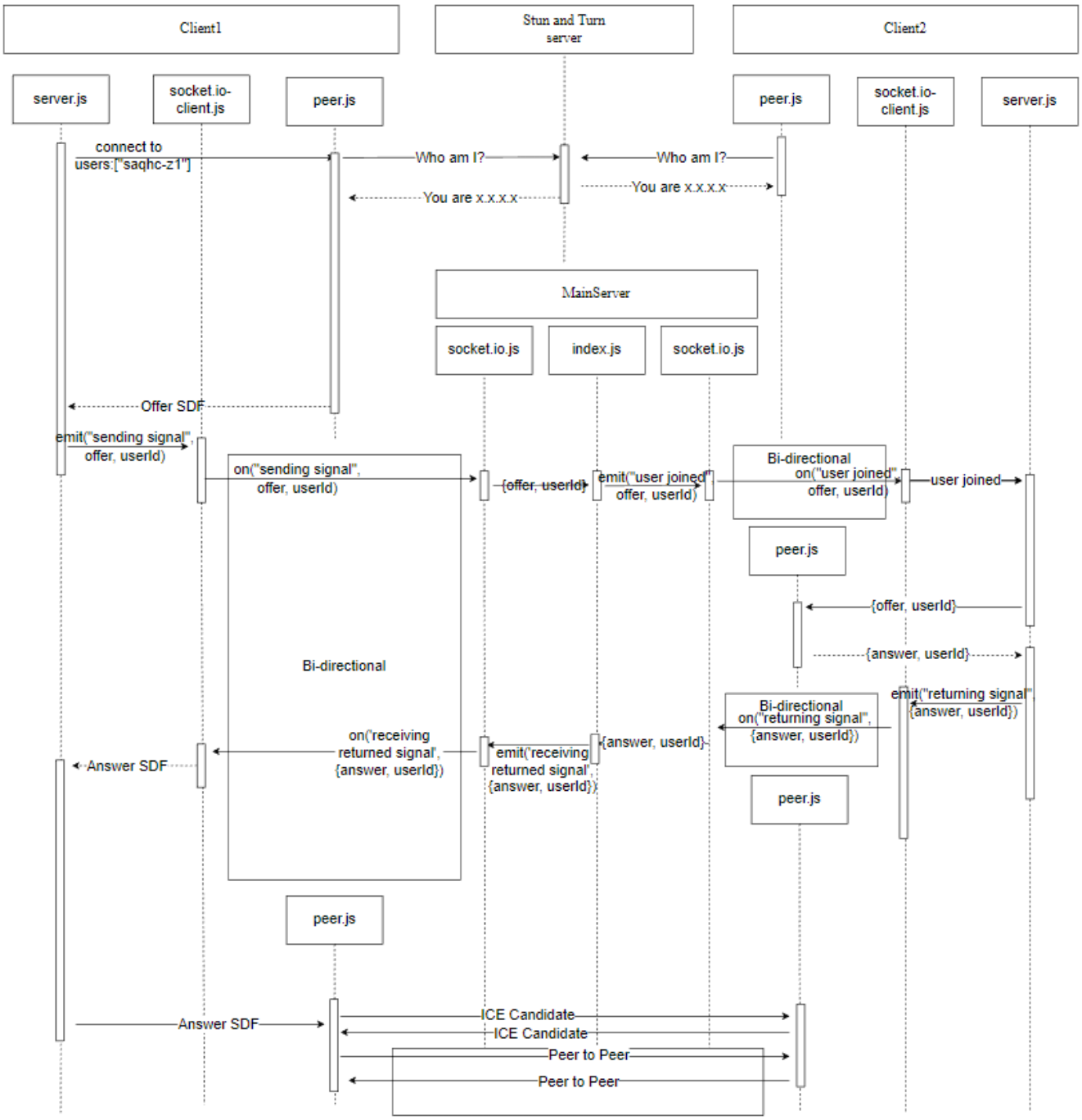

Рисунок 3.6 – Діаграми послідовності прямого підключення та обміну даними між клієнтами

3.3 Модель проектування програмного додатку

3.3.1 Діаграма класів.

Діаграми класів показує структуру моделі, а також містить наступні елементи: класи, типи даних, відображення їх змісту. Дана діаграма показує статичні структури моделі об'єктно-орієнтованого програмування [26].

Діаграми класів програмного додатку зображено на рис. 3.7.

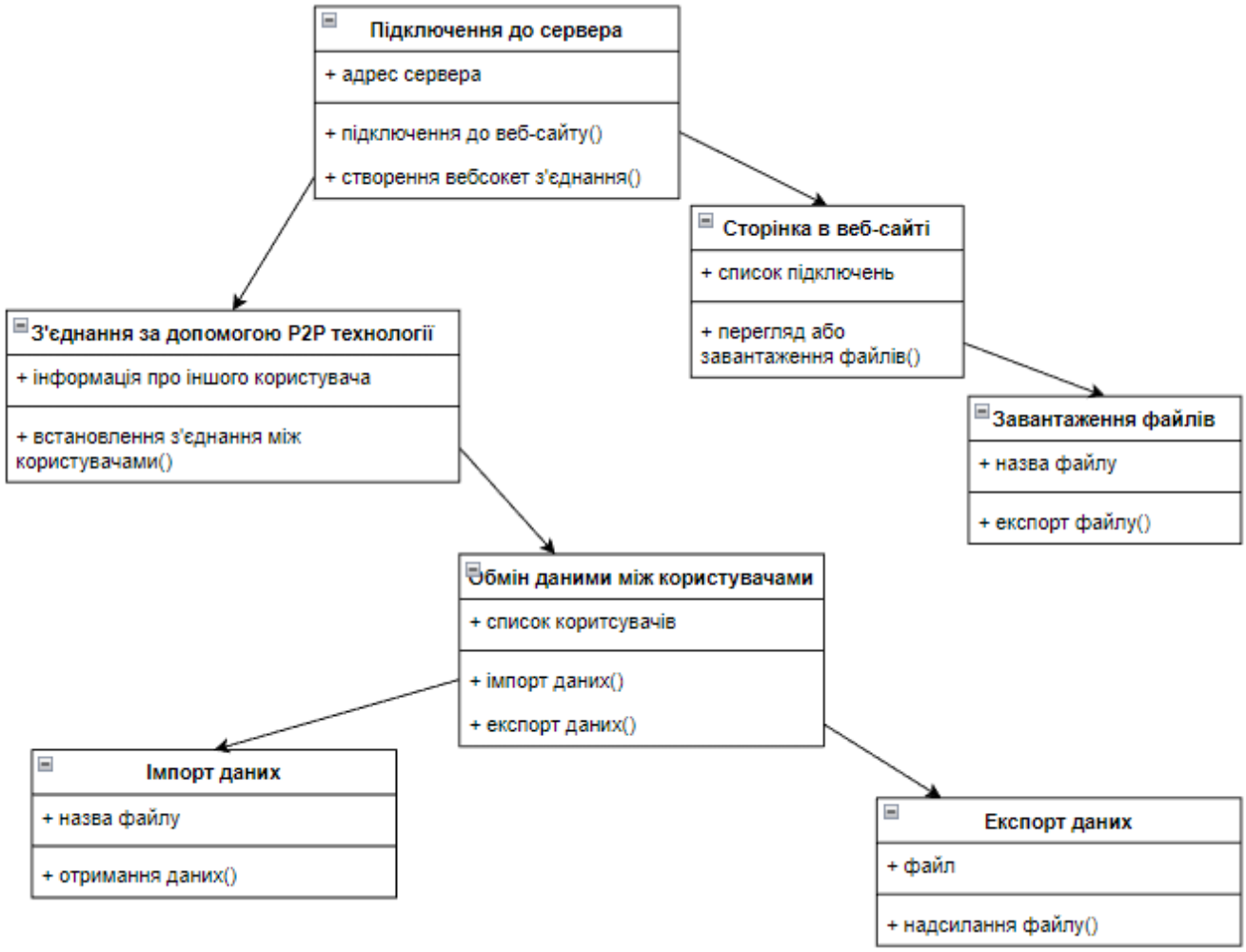

Рисунок 3.7 – Діаграма класів

3.3.2 Діаграми діяльності.

Діаграма діяльності – це діаграма, яка відображає послідовність виконання дій. Вона має формат графа, вершини якого визначають стани видів діяльності чи певних дій, а дуги – переходи від одного стану до іншого [27].

Діаграму діяльності програмного додатку зображено на рис. 3.8.

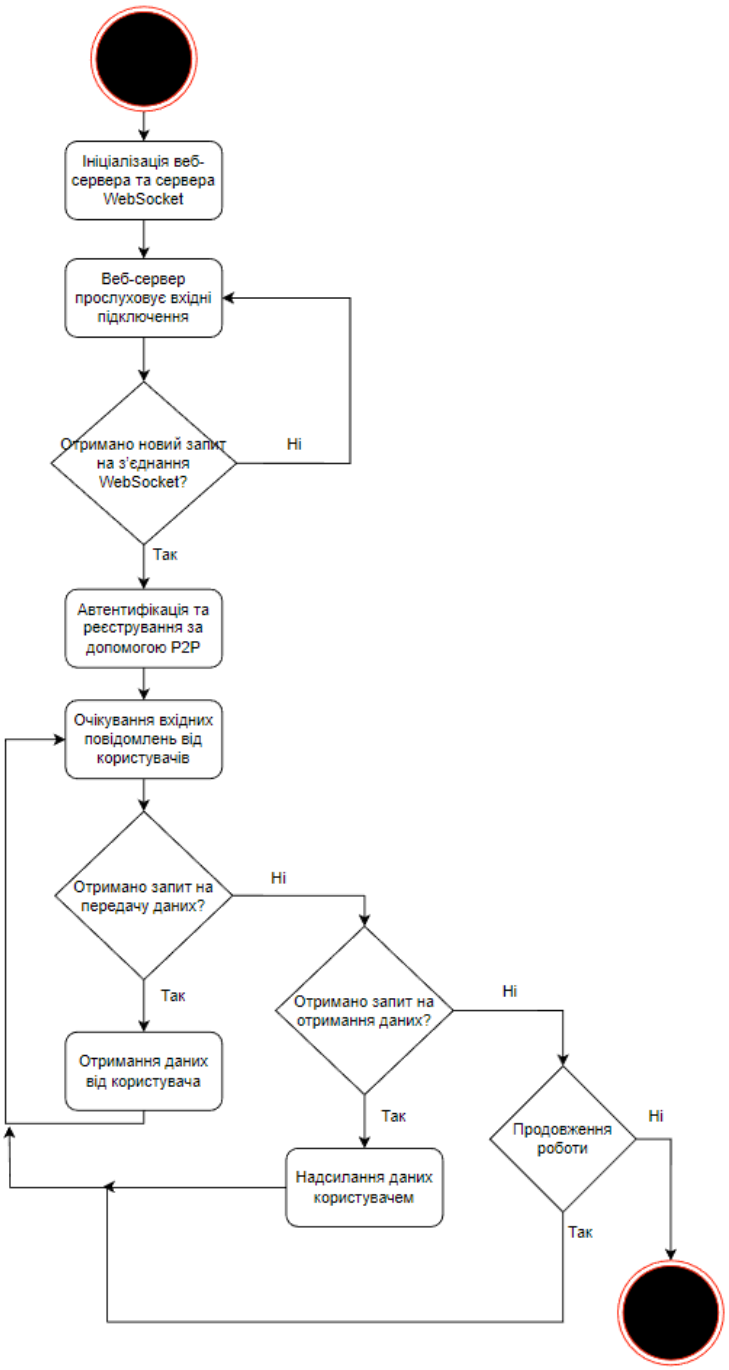

Рисунок 3.8 – Діаграма діяльності

3.3.3 Діаграма варіантів використання.

Діаграма варіантів використання використовується для показу можливих відношень між акторами та прецедентами.

На рис. 3.9 показано можливі варіанти використання хмарного додатку різними користувачами, а саме:

– огляд віртуального середовища з використанням шолому віртуальної реальності;
– підключення до сервера;
– перегляд даних в інтерфейсі на веб-сервері;
– з’єднання з іншими користувачами;
– імпорт даних від іншого користувача;
– експорт даних іншому користувачу.

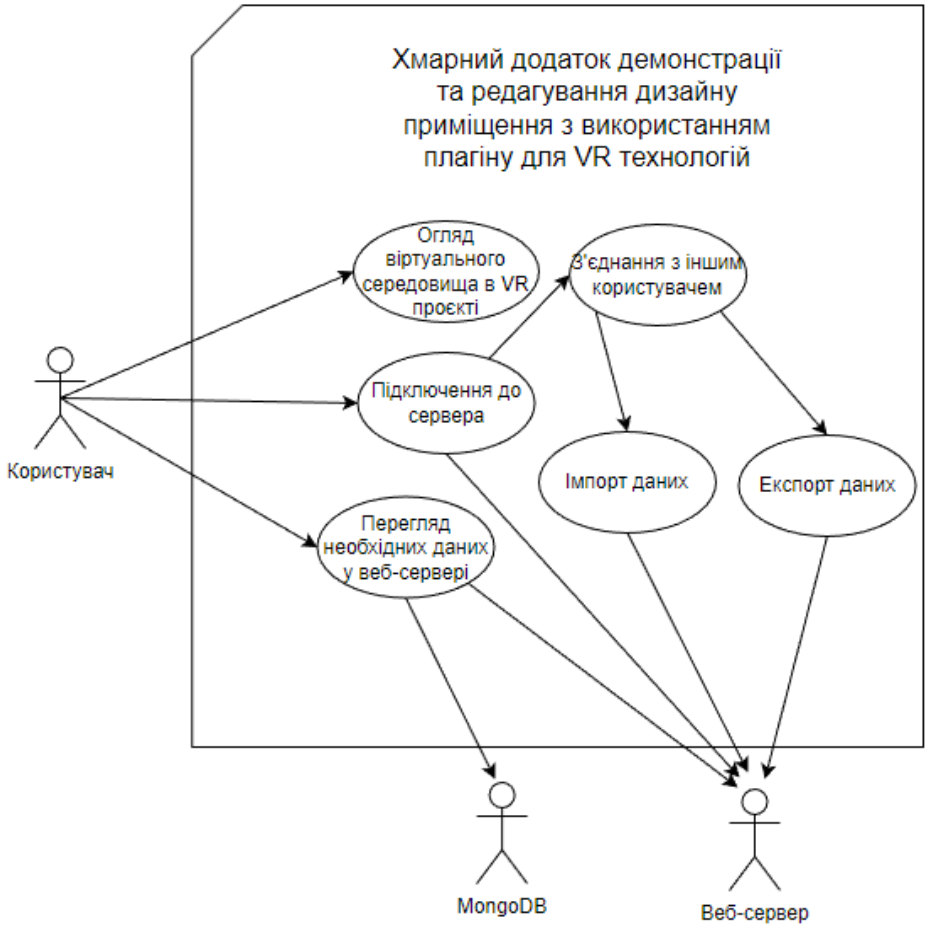

Рисунок 3.9 – Діаграма діяльності

3.5 Модель реалізації

Діаграма розгортання використовується у ході розроблення конфігурації програмної системи. Вона створюється у вигляді графічного зображення процесів, пристроїв і зав’язків між ними. Тим самим дана діаграми підсумовує процес об’єктного програмування і є останнім етапом специфікації моделювання [25].

Діаграма розгортання програмного додатку зображено на рис. 3.10.

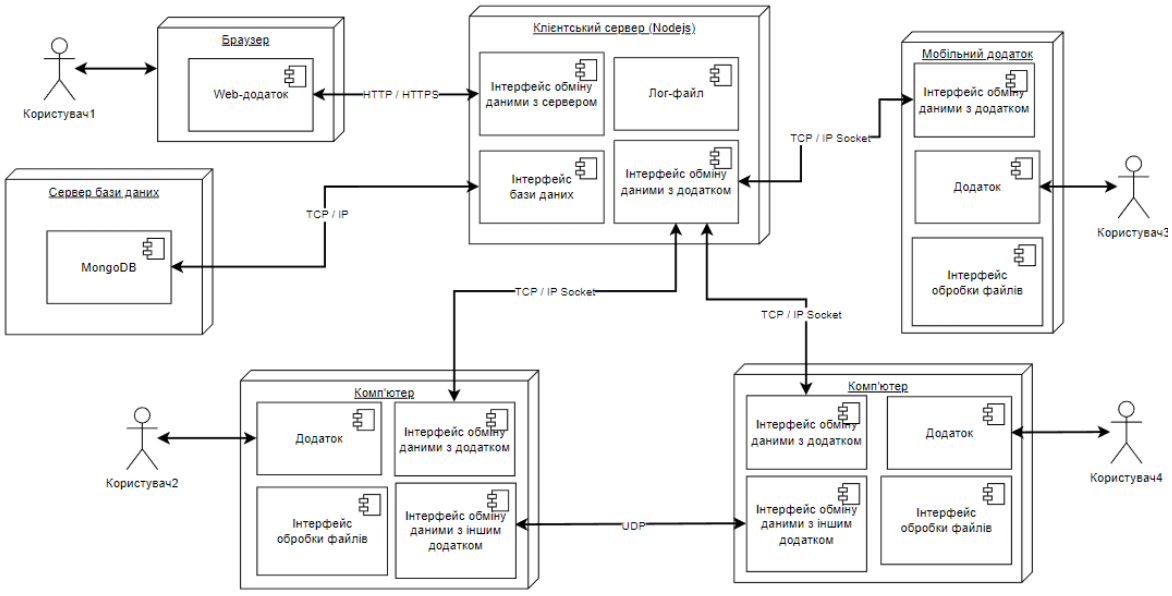

Рисунок 3.10 – Діаграма розгортання програмного додатку

3.6 Моделювання даних

3.6.1. Логічна модель даних

Використання бази даних в даному проекті необхідне для реалізації можливості відслідковувати прогрес передачі даних та цілісність файлів.

На етапі фізичного проектування логічна модель даних є основним джерелом інформації. Вона допомагає виконати аналіз будь-яких змін, що внесені до бази даних [28].

Детальні відомості про атрибути та типи даних показано у таблицях 3.1-3.4.

Таблиця 3.1 – Список всіх сутностей бази даних

User	Користувач
Room	Віртуальна кімната
File	Файл для обміну

Таблиця 3.2 – Користувач (User)

№	Поле	Атрибут	Тип
1	userID	Унікальний ідентифікатор користувача	UUID(36)
2	Name	Ім’я користувача	VARCHAR(255)

Таблиця 3.3 – Віртуальна кімната (Room)

№	Поле	Атрибут	Тип
1	roomID	Унікальний ідентифікатор користувача	UUID(36)
2	Fields	Ім’я користувача	VARCHAR(255)

	UserID	Унікальний ідентифікатор користувача	UUID(36)

Таблиця 3.4 – Файл для обміну (File)

№	Поле	Атрибут	Тип
1	fileID	Унікальний ідентифікатор користувача	UUID(36)
2	fileName	Ім’я користувача	VARCHAR(255)
	chunks	Інформація про частини файлу	OBJECT
	chunks.name	Назва частини файлу	STRING
	chunks.bytes	Розмір частини файлу	NUMBER

Логічна модель даних показано на рис. 3.11

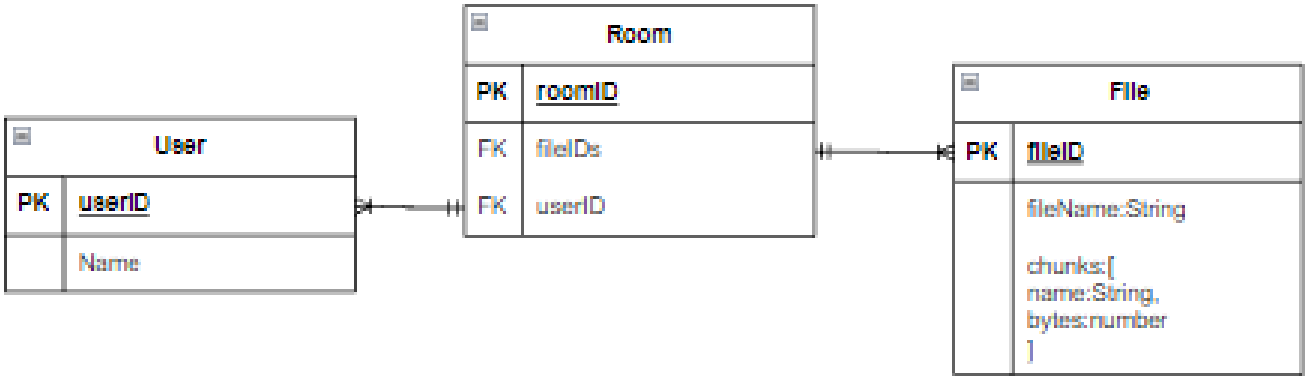

Рисунок 3.11 – Логічна модель даних

В результаті проведеного моделювання додатку було сформовано його структуру та функціональні можливості. На їх основі був проведений вибір інструментів для розробки додатку.

Для реалізації проекту потрібен хостинг, ідеальним варіантом буде використання власного серверу з відкритим доступом до глобальної мережі, бо він дозволить забезпечити прямий доступ до налаштування та тестування роботи

продукту [29]. Для розробки використовуватиметься Visual Studio [30], універсальне середовище розробки з оптимальним набором функцій. В якості шаблону проекту було обрано веб-програма Node.js [31]. Адже її використання дозволить створити сервер, який керуватиме та надаватиме користувачам підключення за допомогою P2P технології [32]. Для роботи серверу буде реалізована база даних MongoDB [33], так як вона не вимагає опису схеми таблиць у порівнянні з реляційними базами даних. Дані будуть зберігатися у вигляді колекцій, де не обов'язково повинна бути схожа структура даних та документів. Для демонстрації підключення клієнтів та обміну даними між ними буде використано середовище Unreal Engine 4 з підтримкою функцій віртуальної реальності [34].

4 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ДОДАТКУ

4.1 Розробка програмного продукту

Для розробки були використані Node Package Manager і також NodeJS, два потужних інструменти, що забезпечують гнучкість і продуктивність у розробці серверної частини програмного забезпечення. Для роботи з базою даних було використано MongoDB, сучасну базу даних NoSQL, що працює з документами у форматі JSON. (рис. 3.10). Розробка проводилася у середовищі Visual Studio Code, яке є одним із найпопулярніших середовищ розробки відкритого коду. Щодо підключення до сервера, було розроблено наступні компоненти, які зображені на рисунку 4.1. Ці компоненти були розроблені таким чином, щоб забезпечити надійне та стабільне підключення до сервера, їх перелік наведений нижче:

- socket.on('loginning', loginningInWeb) встановлює обробник loginningInWeb для події 'loginning'. Ця подія виникає, коли користувач намагається увійти на веб-сайт.
- socket.on('changeName', changeNameInWeb) встановлює обробник changeNameInWeb для події 'changeName'. Ця подія виникає, коли користувач змінює своє ім'я на веб-сайті.
- socket.on('disconnect', webUserDisconnect) встановлює обробник webUserDisconnect для події 'disconnect'. Ця подія виникає, коли користувач відключається від сервера.
- socket.on('getListRooms', getListRooms) встановлює обробник getListRooms для події 'getListRooms'. Ця подія виникає, коли запитується список кімнат на веб-сайті.

– socket.on('createNewRoom', createNewRoom) встановлює обробник createNewRoom для події 'createNewRoom'. Ця подія виникає, коли користувач створює нову кімнату на веб-сайті.
– socket.on('joinToRoom', webUserJoinToRoom) встановлює обробник webUserJoinToRoom для події 'joinToRoom'. Ця подія виникає, коли користувач приєднується до кімнати на веб-сайті.
– socket.on('getRoomInfor', getRoomInfor) встановлює обробник getRoomInfor для події 'getRoomInfor'. Ця подія виникає, коли запитується інформація про певну кімнату на веб-сайті.
– socket.on('sendFileChunk', saveFileChunkFromWeb) встановлює обробник saveFileChunkFromWeb для події 'sendFileChunk'. Ця подія виникає, коли на веб-сайті надсилається шматок файлу.
– socket.on('downloadFileByChunks', sendFileByChunksToWeb) встановлює обробник sendFileByChunksToWeb для події 'downloadFileByChunks'.

```
io.on('connection', socket => {
    console.log("New user connected");
    socket.on('loginning', loginningInWeb);
    socket.on('changeName', changeNameInWeb);
    socket.on('disconnect', webUserDisconnect);
    socket.on('getListRooms', getListRooms);
    socket.on('createNewRoom', createNewRoom);
    socket.on('joinToRoom', webUserJoinToRoom);
    socket.on('getRoomInfor', getRoomInfor);
    socket.on('sendFileChunk', saveFileChunkFromWeb);
    socket.on('downloadFileByChunks', sendFileByChunksToWeb);
    socket.on('openAccessToTransferByP2P', openAccessToTransferByP2P);

    socket.on('offerToUser', offerToUser);
    socket.on('answerToUser', answerToUser);
    socket.on('iceCandidateToUser', iceCandidateToUser);
```

Рисунок 4.1 Підключення до сервера

Для можливості його використання потрібна підтримка вебсокетів з боку плагіну. Вебсокети дозволяють двостороннє комунікаційне з'єднання між клієнтом і сервером, що відкриває можливості для реалізації в реальному часі. Додатково, для прямого підключення до інших користувачів потрібно мати підтримку інтернет-протоколу WebRTC. Він дозволяє здійснювати пряме підключення між браузерами або програмами для обміну потоковими даними, включаючи аудіо, відео та просто даними.

Реалізація оновлення в період часу 3 секунди для статусу обміну файлами показана на рисунку 4.2.

```
setInterval(checkWriteStreamFiles, 3000);

function checkWriteStreamFiles() {
    if (streamFiles.length != 0) {
        const roomWithFileTransferInformation = streamFiles.reduce((array, writeStreamFile) => {
            if (array[writeStreamFile.roomId] == null) {
                array[writeStreamFile.roomId] = [];
            }
            array[writeStreamFile.roomId].push(writeStreamFile.fileTransferInformation);
            return array;
        }, {});
        for (const roomName in roomWithFileTransferInformation) {
            const fileTransferInformations = roomWithFileTransferInformation[roomName].reduce((array, fileT
                if (array == null) {
                    array = [];
                }
                array.push(fileTransferInformation.getObject());
                return array;
            }, []);
            const users = rooms[roomName].users;
            users.forEach(user => {
                io.to(user.socketId).emit('emitFileTransferInformationsInRooms', { error: false, fileTransf
            })
        }
        console.log("test");
    }
}
```

Рисунок 4.2 Оновлення статусу обміну файлами

Для відображення веб-сторінки було розроблено сервер, що відповідає за обробку даних. Приклад реалізації показано на рисунку 4.3.

```
const express = require("express");

const http = require("http");

const path = require('path');

const { engine } = require('hbs');

const app = express();

const server = http.createServer(app);

app.set("view engine", "hbs");
app.set('views', path.join(__dirname, 'views'));
app.set('view options', { layout: 'layouts/main' });

app.use(express.static(path.join(__dirname, 'public')));

app.get('/', function (req, res) {
    res.render('index.hbs' );
});

app.get('/rooms/:roomName', function (req, res) {
    res.render('room.hbs');
});


server.listen(8000, () => console.log('server is running on port 8000'));

module.exports = {
    server: server,
    express: express
}
```

Рисунок 4.3 Сервер для веб-сторінки

Було реалізовано підключення до серверу з використанням веб-сокету. Приклад реалізації показано на рисунку 4.4.

```
function connectToServer(nameError, userNameInput) {
  return new Promise((resolve, reject) => {
    var socket = io();
    socket.on('connect', () => {
      document.getElementById('serverInformation').textContent = 'Встановлено звязок з сервером';
      nameError.classList = "text-info"
      nameError.innerHTML = 'Вхід до сервера'
      socket.emit("loginning");
    });
    socket.on('reconnect', () => {
      console.log('Socket reconnected');
      window.location.reload();
    });
    socket.on('disconnect', () => {
      document.getElementById('serverInformation').textContent = 'Втрачено звязок з сервером';
      if (confirm('Втрачено звязон з сервером зробити перезагрузку?')) {
        window.location.reload();
      } else {
        window.location.href = '/';
      }
    });
    socket.on('emitLoginning', (data) => {
      userNameInput.value = data.userName;
      sessionStorage.setItem('userName', data.userName);
      nameError.innerHTML = 'Вхід як користувач завершено'
      console.log(data);
      resolve(socket);
    });
    socket.on("emitChangeName", (data) => {
      if (!data.error) {
        userNameInput.value = data.userName;
        sessionStorage.setItem('userName', data.userName);
        nameError.innerHTML = 'Імя змінено';
        nameError.classList = "text-success"
      } else {
        nameError.style.display = 'block'
        nameError.classList = "text-danger"
        nameError.innerHTML = data.message
      }

    });
  });
}

function emitToCheckUserName(userName) {
  socket.emit('userExsist', userName);
}
```

Рисунок 4.4 Підключення через веб-сокет

Далі було роблено логіку прямого підключення та можливість завантаження файлів для користувачів. Приклад реалізації показано на рисунку 4.5.

```
function connectToUserAndDownloadFile(socket, toUserName, roomId, fileName) {
    const peerConnection = new RTCPeerConnection({
        iceServers: [
            {
                urls: 'turn:relay1.expressturn.com:3478',
                username: 'efTL13SHG2VNFCIEIX',
                credential: '8n1GZ7ljeyZIgSNg'
            },
            {
                urls: 'stun:stun.l.google.com:19302'
            }
        ]
    });

    peerConnection.onnegotiationneeded = async () => {
        const offer = await peerConnection.createOffer();
        await peerConnection.setLocalDescription(offer);

        socket.emit('offerToUser', {
            offer: offer,
            from: sessionStorage.getItem('userName'),
            to: toUserName,
            roomId: roomId,
            fileName: fileName
        });
    };

    let receivedChunks = [];
    let dataChannel = peerConnection.createDataChannel('fileTransfer');


    socket.on('emitAnswerToUser', async (data) => {
        if (!data.error) {
            await peerConnection.setRemoteDescription(data.answer);
        } else {
            nameError.style.display = 'block'
            nameError.classList = "text-danger"
            nameError.innerHTML = data.message
        }
    });

    peerConnection.onicecandidate = ({ candidate }) => {
        if (event.candidate) {
            console.log('ICE Candidate:', event.candidate);
        } else {
            console.log('ICE Candidate gathering complete.');
        }
```

Рисунок 4.5 Реалізація прямого підключення

Для взаємодії між користувачами було розроблено канал для прийняття та передачі файлів. Приклад реалізації показано на рисунку 4.6.

```
function createDataChannel(peerConnection, receivedChunks, fileName) {
    const dataChannel = peerConnection.createDataChannel('fileTransfer');
    dataChannel.onmessage = (event) => {
        const message = event.data;

        if (message === 'EOF') {
            const fileBlob = new Blob(receivedChunks);
            const fileURL = URL.createObjectURL(fileBlob);
            const downloadLink = document.createElement('a');
            downloadLink.href = fileURL;
            downloadLink.download = fileName;
            downloadLink.click();
            URL.revokeObjectURL(fileURL);
        } else {
            // Concatenate the received chunk to the binary string
            receivedChunks.push(message);
            event.target.send("ok");
        }
    };
}
```

Рисунок 4.6 Канал обміну файлами

Для передачі файлів було створено алгоритм прямого підключення між користувачами. Приклад реалізації показано на рисунках 4.7-4.8.

```
async function connectToUserAndSendFile(socket, offer, toUserName, file) {
    const peerConnection = new RTCPeerConnection({
        iceServers: [
            {
                urls: 'turn:relay1.expressturn.com:3478',
                username: 'efTL13SHG2VNFCIEIX',
                credential: '8n1GZ71jeyZIgSNg'
            },
            {
                urls: 'stun:stun.l.google.com:19302'
            }
        ]
    });

    await peerConnection.setRemoteDescription(offer);

    const answer = await peerConnection.createAnswer();
    await peerConnection.setLocalDescription(answer);

    socket.emit('answerToUser', {
        answer: answer,
        roomId: roomId,
        from: sessionStorage.getItem('userName'),
        to: toUserName
    });

    socket.on('emitIceCandidateToUser', (data) => {
        if (!data.error) {
            peerConnection.addIceCandidate(data.candidate);
        } else {
            nameError.style.display = 'block'
            nameError.classList = "text-danger"
            nameError.innerHTML = data.message
        }
    });

    var fileBinaryString = null;
    let i = 0;
```

Рисунок 4.7 Створення прямого підключення

```
peerConnection.ondatachannel = ({ channel }) => {
    console.log('Data channel:', channel);
    channel.onmessage = (event) => {
        if (i < fileBinaryString.length) {
            channel.send(fileBinaryString[i]);
            console.log((i / fileBinaryString.length) * 100 + "%");
            i++;
        } else {
            channel.send('EOF');
        }
    };
    channel.onopen = async () => {
        fileBinaryString = await readFileAsync(file);
        channel.onbufferedamountlow = () => {
            console.log('Send queue is no longer full. You can resume sending data.');
        };
        if (i < fileBinaryString.length) {
            channel.send(fileBinaryString[i]);
            i++;
        } else {
            channel.send('EOF');
        }
    };
};
```

Рисунок 4.8 Продовження рисунка 4.7

Для демонстрації роботи з’єднання з Unreal Engine була реалізовані необхідні методи для отримання повідомлень від серверу за допомогою веб-сокету. Зображення алгоритму показано на рисунку 4.9-4.11.

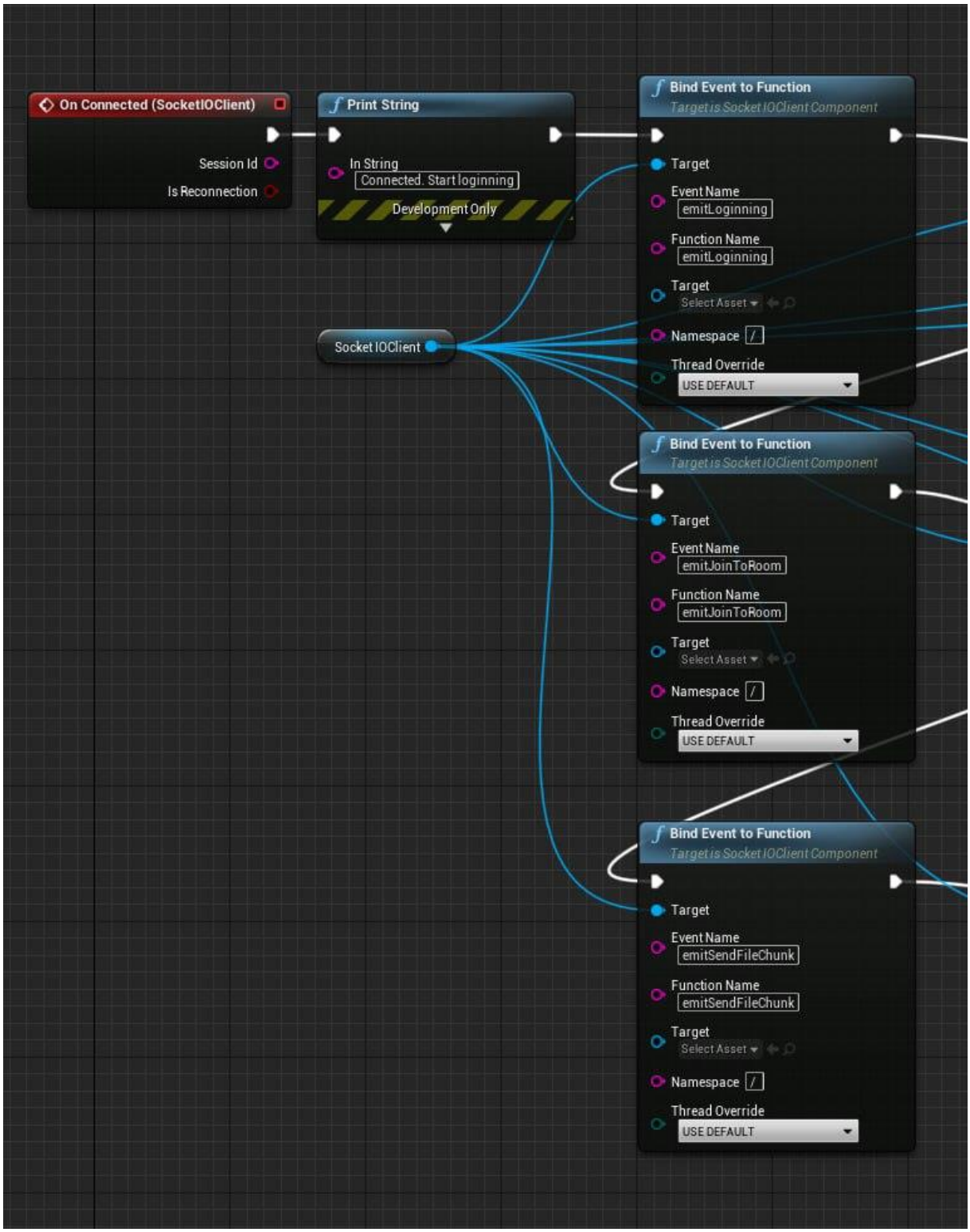

Рисунок 4.9 Реалізація підключення в Unreal Engine

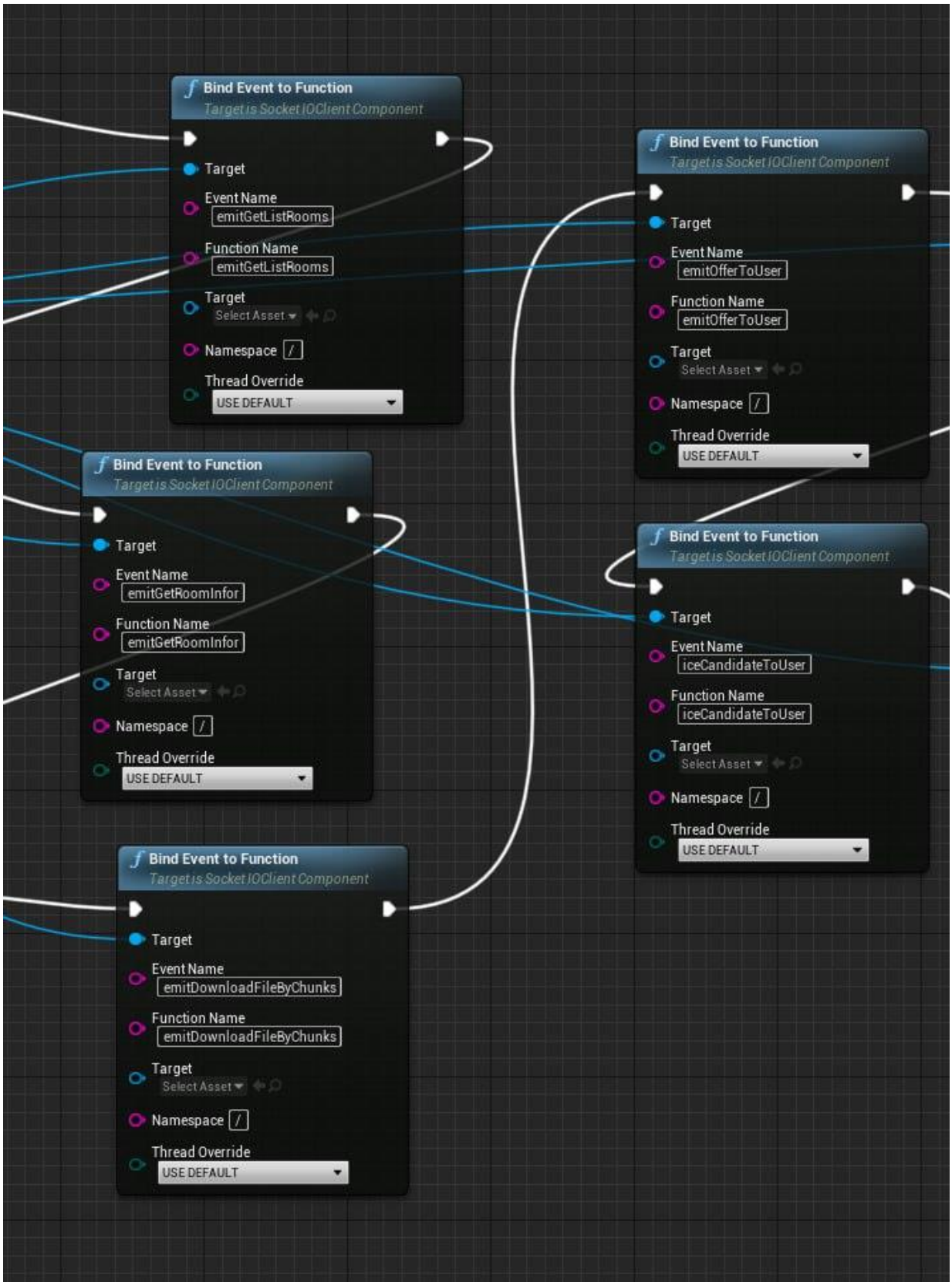

Рисунок 4.10 Продовження рисунка 4.9

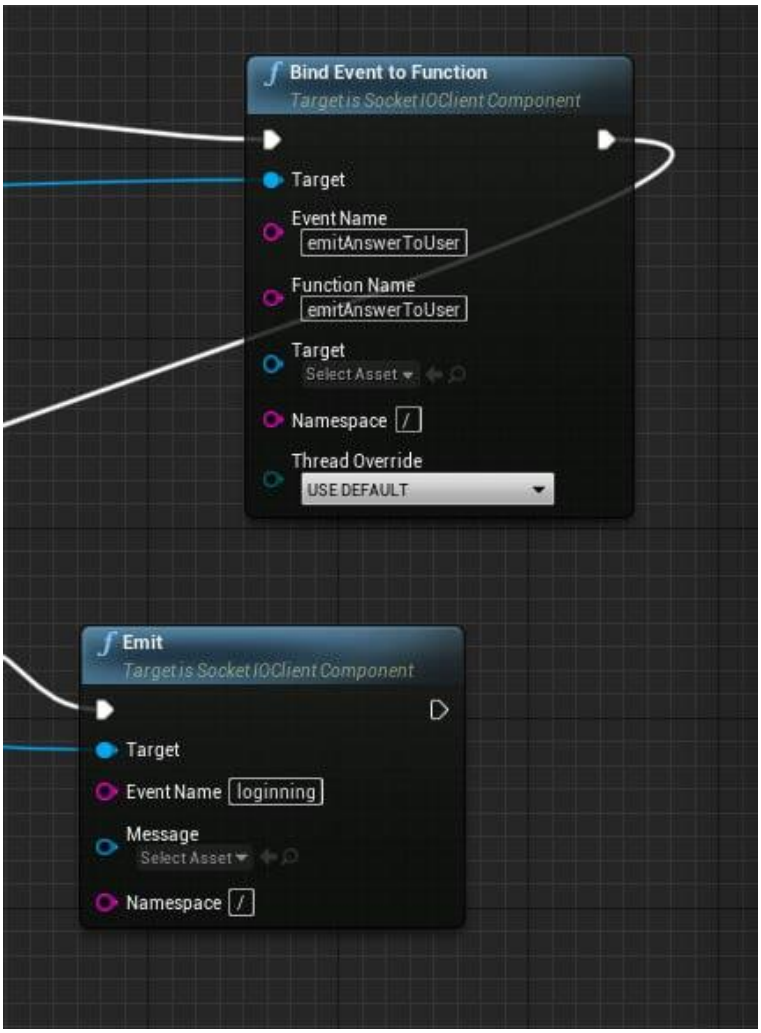

Рисунок 4.11 Продовження рисунка 4.10

Налаштування прямого підключення в Unreal Engine виконано наступним чином:

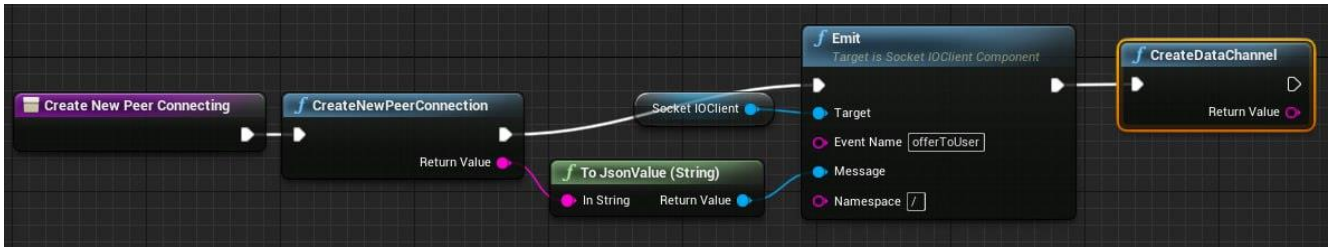

Рисунок 4.12 Алгоритм прямого підключення

Створення класу для обміну файлами показано на рисунку 4.13.

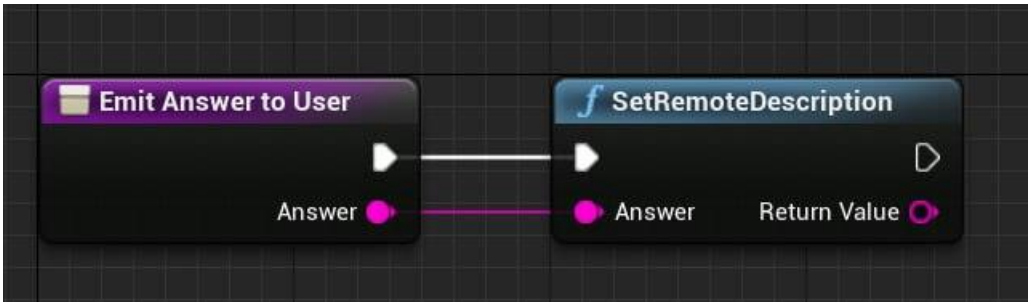

Рисунок 4.13 Клас для обміну даними

4.2 Дослідження характеристик передачі даних

Було проведено дослідження таких характеристик передачі даних між сервером та клієнта, а також між клієнтами:

- стабільність підключення;
- стабільність передачі даних;
- поведінка сервера у випадку розриву з’єднання з користувачем;
- час відгуку серверу.

Проведення дослідження відбувалося на сервері з вказаними у таблиці 4.1 технічними характеристиками:

Таблиця 4.1 Характеристики обладнання

Обладнання	Детальна інформація
Операційна система	Ubuntu 20.04 LTS
Процесор	ARM Cortex-A72
RAM	8 ГБ LPDDR4 SDRAM
Мережеве з'єднання	WIFI

Було створено план тестування, а саме:

– створення підключення;
– передача даних;
– відключення;
– масштабування.

Результат проведених тестів за вказаними вище пунктами показано на таблиці 4.2.

Таблиця 4.2 Результати тестування

Створення підключення	
Сценарій	Результат
Було проведено декілька тестів з метою перевірки швидкості та надійності створення підключення з використанням технології PeerToPeer та SocketIO для налаштування підключення. Для цього було використано два вузли, які знаходились в різних мережах. Проведено тестування за допомогою автоматизованого скрипту, який намагався встановити з'єднання між вузлами	За результатами тестування, середній час створення підключення становив приблизно 150 мс, що відповідає очікуванням для системи цього типу. Всі підключення було успішно встановлено без помилок. Не було виявлено ніяких проблем з стабільністю підключення чи його перериваннями

Таблиця 4.3 Результати тестування

Передача даних	
Сценарій	Результат
Вузол А встановлює підключення до вузла В	Було встановлено підключення між вузлами А і В
Вузол А відправляє текстове повідомлення вузлу В	Усі повідомлення, які відправляє вузол А, успішно отримуються вузлом В без будь-яких втрат або пошкоджень
Вузол В підтверджує отримання повідомлення, відправляючи відповідь вузлу А	Вузол В успішно відправляє відповідь вузлу А, підтверджуючи отримання кожного повідомлення
Повторення попередніх сценаріїв кілька разів, змінюючи розмір повідомлення та швидкість передачі	Час відправлення та отримання повідомлень залишається стабільним незалежно від розміру повідомлення та швидкості передачі

Таблиця 4.4 Результати тестування

Відключення	
Сценарій	Результат
Відключення одного з вузлів та відправка інформації про це всім іншим вузлам	При відключенні користувача, система вірно розсилає повідомлення про це всім іншим користувачам в мережі. Цей процес відбувається без затримок та помилок. Також було визначено, що після відключення користувача, система звільняє ресурси, що були пов'язані з цим користувачем, що позитивно впливає на загальну продуктивність системи

Таблиця 4.5 Результати тестування

Масштабування	
Сценарій	Результат
Встановлення 1000 одночасних підключень	За наявності 8 ГБ оперативної пам’яті, система забезпечує стабільну роботу до приблизно 700 одночасних підключень. Після цього моменту, з початком перевантаження пам'яті, час відгуку починає зростати і спостерігаються випадки втрати підключень. Це свідчить про необхідність додаткового масштабування ресурсів або оптимізації використання пам'яті для підтримки більшої кількості підключень

На основі проведеного тестування, система PeerToPeer на базі SocketIO продемонструвала стабільну роботу на всіх стадіях: від створення підключення до його завершення, включаючи передачу даних, обробку відключень та масштабування.

Всі створені підключення були успішно встановлені без помилок, а середній час створення підключення відповідав очікуванням та становив близько 150 мс. Передача даних також була ефективною, з усіма повідомленнями, які були відправлені та отримані без втрат або пошкоджень.

Система також виявилася ефективною при відключенні користувачів, правильно відправляючи повідомлення про відключення всім іншим користувачам та вільно відповідаючи ресурси, пов'язані з відключеним користувачем.

Однак, під час тестування масштабування, було виявлено, що при більше ніж 700 одночасних підключень, система починає перевантажуватись, що веде до збільшення часу відгуку та випадків втрати підключень. Це свідчить про те, що для більшого обсягу підключень, потребується або більше ресурсів, або оптимізація використання пам'яті.

ВИСНОВКИ

У ході виконання дипломної роботи було проведено аналіз предметної області, дослідження наявних програмних продуктів-аналогів. На основі отриманих даних було сформовано мету та задачі дослідження.

Було виконано проектування додатку. Проведення моделювання допомогло визначити об’єкти, функції та процеси, що відбуваються у програмному додатку. Для цього були виконані наступні етапи:

- моделювання програмного додатка в нотації IDEF0;
- створено модель аналізу предметної області програмного додатку;
- створено модель проектування програмного додатку;
- створено модель реалізації;
- проведено моделювання даних.

На основі розділу проектування було обрано основні інструменти для реалізації програмно додатку, а саме:

- Visual Studio;
- Node.js;
- P2P технологія підключення;
- база даних MongoDB;
- Unreal Engine 4.

Для реалізації програмного додатку було виконано наступні кроки:

- створення серверу за допомого фреймворка Node.js, на якому буде реалізована взаємодія з веб-сокетами;
- створення бази даних для серверу для зберігання всієї необхідної інформації;
- створення веб-сторінки для серверу з ціллю відображення необхідних даних стосовно підключення клієнтів;

- створення плагіну для Unreal Engine 4 для демонстрації роботи розробленої технології;
- розміщення проекту на хостингу;
- дослідження основних характеристик процесу передачі даних між клієнтами.

Результатом роботи є досягнення поставленої мети, а саме було розроблено програмне забезпечення для обміну даними між додатками, що працюють на базі різних операційних систем, яке містить серверну частину для виконання основних процесів та плагін для демонстрації роботи системи в середовищі Unreal Engine

Розроблений додаток містить універсальний метод обміну файлами шляхом прямого підключення, що дозволяє обмінюватися даними між додатками або сервісами на базі різних платформ без використання додаткових серверів, які обробляють проміжні дані, тим самим зменшуючи залежність від сторонніх постачальників обслуговування і потенційні ризики для конфіденційності даних користувача.

В результаті проведеного дослідження розробленого програмного додатку виявилися деякі недоліки, а саме наразі обмежена швидкість передачі даних при великій кількості одночасно підключених користувачів, проте поточної пропускної здатності достатньо для роботи з файлами, які використовуються в даному випадку.

До плану майбутнього покращення додатку можна віднести наступні дії:

- враховуючи високу завантаженість при більше ніж 700 одночасних підключеннях, необхідно провести оптимізацію коду для ефективнішого використання ресурсів;
- підвищення конфіденційності даних користувачів;
- для забезпечення більш стабільності та продуктивності при великому навантаженні можна розглянути можливість горизонтального масштабування, додавши додаткові сервери до системи.

Результати роботи пройшли апробацію на конференції IMA:2023.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Технології передачі даних в комп'ютерній мережі [Електронний ресурс] – Режим доступу до ресурсу: http://pro-computer.pp.ua/5404-tehnologyi-peredach-danih-v-kompyuterny-merezh.html.

2. Virtual Reality (VR) [Електронний ресурс] – Режим доступу до ресурсу: https://it-enterprise.com/knowledge-base/technology-innovation/virtualnaja-realnost-vr?preview=1.

3. VR Applications: 23 Industries using Virtual Reality [Електронний ресурс] – Режим доступу до ресурсу: https://virtualspeech.com/blog/vr-applications.

4. Технології передачі даних в комп'ютерній мережі [Електронний ресурс] – Режим доступу до ресурсу: http://pro-computer.pp.ua/5404-tehnologyi-peredach-danih-v-kompyuterny-merezh.html.

5. File transfer [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://www.ibm.com/topics/file-transfer.

6. Kurose J. Computer Networking: A Top-Down Approach (7th ed.). Pearson. / J. Kurose, K. Ross., 2017.

7. Постковідний світ та технології: як пандемія вплинула на стартап-екосистему [Електронний ресурс] – Режим доступу до ресурсу: https://www.epravda.com.ua/columns/2021/01/19/670136/.

8. Mendeley [Електронний ресурс] – Режим доступу до ресурсу: https://www.mendeley.com/search/?page=2&publicationYear=2022&query=p2p%20connection&sortBy=relevance.

9. Lloret J, Sendra S, Jemenez J, Parra L, “Providing security and fault tolerance in P2P connections between clouds for mHealth services”, vol. 9, 2016, doi: 10.1007/s12083-015-0378-3

10. Kim E, Kim J, Lee C, “Efficient neighbor selection through connection switching for P2P live streaming”, vol.10, 2019, doi: 10.1007/s12652-018-0691-9

11. Drnasin I, Grgic M, Gledec G, “Exploring WebRTC Potential for DICOM File Sharing”, vol. 33, 2020, doi: 10.1007/s10278-019-00305-0

12. Ahmadi M, Pocta P, Melvin H, “An adaptive bitrate switching algorithm for speech applications in context of webrtc”, vol. 17, 2021, doi: 10.1145/3458751

13. Gortázar F, Gallego M, Maes-Bermejo M, “Cost-effective load testing of WebRTC applications”, vol. 193, 2022, doi: 10.1016/j.jss.2022.111439

14. García B, Gallego M, Gortázar F, “Understanding and estimating quality of experience in WebRTC applications”, vol. 101, 2019, doi: 10.1007/s00607-018-0669-7

15. Tang D, Zhang L, “Audio and Video Mixing Method to Enhance WebRTC”, vol. 8, 2020, doi: 10.1109/ACCESS.2020.2985412

16. Jang-Jaccard J, Nepal S, Celler B, “WebRTC-based video conferencing service for telehealth”, vol. 98, 2016, doi: 10.1007/s00607-014-0429-2

17. Patel V, Li C, Rye V, “A Comparison of WebRTC and Conventional Videoconferencing for Synchronized Remote Medical Image Presentation”, vol. 35, 2022, doi: 10.1007/s10278-021-00544-0

18. Cinar Y, Pocta P, Chambers D, “Improved Jitter Buffer Management for WebRTC”, vol. 17, 2021, doi: 10.1145/3410449

19. Feher B, Sidi L, Shabtai A, “WebRTC security measures and weaknesses”, vol.8, 2018, doi: 10.1504/IJITST.2018.092138

20. Ajdinović N, Nurkić S, Baraković Husić J, “Recognition of traffic generated by WebRTC communication”, vol. 1, 2021, doi: 10.54327/set2021/v1.i1.8

21. The streaming torrent client for Mac, Windows, & Linux [Електронний ресурс] – Режим доступу до ресурсу: https://webtorrent.io/.

22. Dat Foundation [Електронний ресурс] – Режим доступу до ресурсу: https://datproject.org/.

23. Sharedrop [Електронний ресурс] – Режим доступу до ресурсу: https://www.sharedrop.io/rooms/abc.

24. Create IDEF0 diagrams [Електронний ресурс] – Режим доступу до ресурсу: https://support.microsoft.com/en-us/office/create-idef0-diagrams-ea7a9289-96e0-4df8-bb26-a62ea86417fc.

25. Unified Modeling Language (UML) | Activity Diagrams [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/unified-modeling-language-uml-activity-diagrams/.

26. Create a UML class diagram [Електронний ресурс] – Режим доступу до ресурсу: https://support.microsoft.com/en-us/office/create-a-uml-class-diagram-de6be927-8a7b-4a79-ae63-90da8f1a8a6b.

27. Create a UML activity diagram [Електронний ресурс] – Режим доступу до ресурсу: https://support.microsoft.com/en-us/office/create-a-uml-activity-diagram-19745dae-2872-4455-a906-13b736f01685.

28. What is a Database Model [Електронний ресурс] – Режим доступу до ресурсу: https://www.lucidchart.com/pages/database-diagram/database-models..

29. Raspberry Pi 4 [Електронний ресурс] – Режим доступу до ресурсу: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/.

30. Visual Studio documentation [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/visualstudio/windows/?view=vs-2022.

31. About documentation [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/en/docs.

32. What's a Peer-to-Peer (P2P) Network? [Електронний ресурс] – Режим доступу до ресурсу:

https://www.computerworld.com/article/2588287/networking-peer-to-peer-network.html.

33. MongoDB Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/docs/.

34. Unreal Engine 4 Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unrealengine.com/4.27/en-US/.

Додаток А

Деталізація мети проекту методом SMART

Продуктом даного проекту є програмний продукт вигляді хмарного додатку, для налагодження зв’язку між додатками, що працюють як на однаковій, так і на різних операційних системах з використанням плагінів і демонстрація його роботи на прикладі VR додатку на базі операційної системи Windows.

Результати деталізації методом SMART наведено у табл. А.1.

Таблиця А.1 − Деталізація мети методом SMART

Specific (конкретна)	Розробити програмний продукт у вигляді хмарного додатку, який допоможе налагодити зв’язок між додатками, що працюють як на однаковій, так і на різних операційних системах з використанням плагінів і продемонструвати його роботу на прикладі VR додатку на базі операційної системи Windows
Measurable (вимірювання)	Ціллю даної роботи є створення хмарного додатку, який спростить процес передачі та обміну даних між пристроями завдяки використанню технології прямого підключення

Таблиця А.2 – Продовження таблиці А.1

Achievable (досяжна, узгоджена)	Ціль даного проекту можна вважати досяжною, адже розробник має навички роботи з WebRTC, а саме технологією прямого підключення, фреймворками NodeJS, SocketIO, JavaScript
Relevant (реалістична)	Для успішної реалізації програмного продукту є відповідні технічні і програмні ресурси, такі як комп'ютер, Raspberry Pi для запуску власного серверу, Visual Studio, доступ до мережі Інтернет
Time-framed (обмежена в часі)	Розробка додатку враховує терміни обмеження в часі, які визначені в календарному плані та матриці відповідальності.

Планування змісту робіт

Для більш ефективного та швидкого виконання робіт, проект можна розбити на менші, контрольовані частини. Для цього можна використати інструмент WBS. За допомогою структури декомпозиції, WBS включає в себе всі необхідні етапи створення проекту та розбиває їх на керовані підрозділи. Кожен етап можна поділити на більш прості завдання, щоб забезпечити можливість їх ефективного виконання.

WBS-структура проекту зображено на рис. А.1.

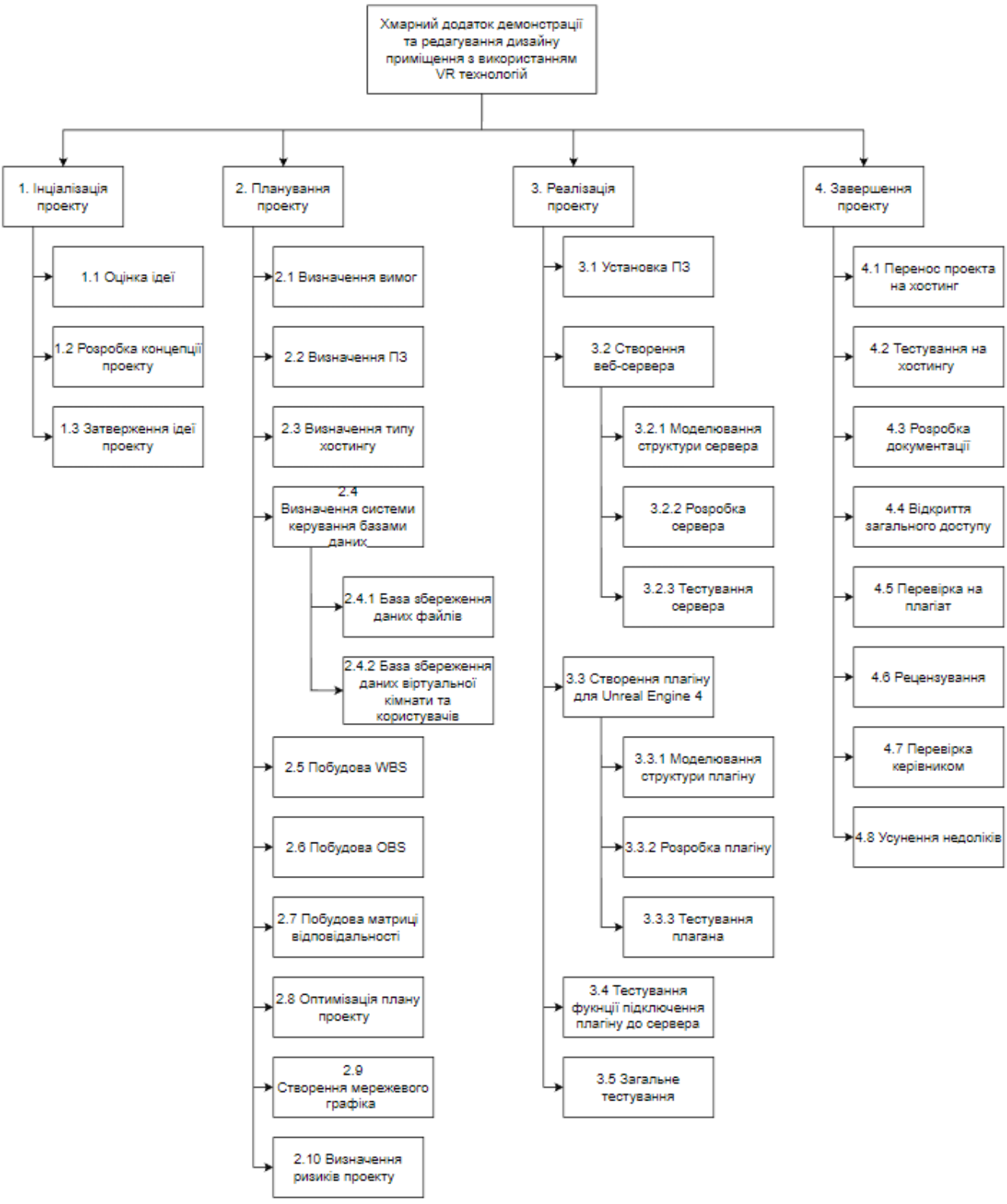

Рисунок А.1 WBS-структура проекту

Планування структури виконавця для впровадження готового проекту

Після складання WBS проводиться розробка організаційної структури виконавців. OBS включає всіх учасників, враховуючи їх ролі та ієрархічні відносини, які беруть участь у проекті. Потім для кожної гілки WBS структури визначаються виконавці на нижньому рівні робіт.

Детальна інформація про виконавців проекту наведена в таблиці А.3.

Таблиця А.3 – Виконавці проекту

Роль	Ім’я	Проектна роль
Програміст	Толстоноженко С.О.	Проводить розробку та тестування основних функції проекту
Дипломний керівник	Шендрик В.В.	Контролює дотримання термінів, виконує збір та аналіз даних
Організаційний менеджер	Толстоноженко С.О.	Створює документацію і плани виконання задач проекту

OBS-структура проекту зображена на рис. А.2.

Рисунок А.2 – OBS - структура проекту

Побудова матриці відповідальності

Створення матриці відповідальності на основі WBS і OBS структур проекту дозволяє призначити відповідальних виконавців для кожного етапу проекту. Це забезпечує контроль за відповідністю розподілу ролей та завдань проекту.

Матрицю відповідальності проекту показано на рисунку А.3.

WBS \ OBS			Толстоноженко	Шендрик
1	1.1			О
	1.2		О	У
	1.3		У	О
2	2.1		О	У
	2.2		О	
	2.3		О	
	2.4	2.4.1	О	
		2.4.2	О	
	2.5		О	
	2.6		О	
	2.7		О	
	2.8		О	У
	2.9		О	У
	2.10		О	У
3	3.1		О	
	3.2	3.2.1	О	
		3.2.2	О	
		3.2.3	О	
	3.3	3.3.1	О	
		3.3.2	О	
		3.3.3	О	
	3.4		О	
4	4.1		О	
	4.2		О	
	4.3		О	У
	4.4		О	
	4.5		У	О
	4.6		О	
	4.7			О
	4.8		О	
О - основна робота; У - приймає участь				

Рисунок А.3 - Матриця відповідальності

Розробка PDM мережі

Для того, щоб уникнути затримок та забезпечити своєчасне завершення проекту, необхідно відвести достатньо часу на побудову PDM-мережі. Ця мережа допомагає у плануванні та відстеженні запланованих задач, та відображає послідовність їх виконання за допомогою стрілок на діаграмі.

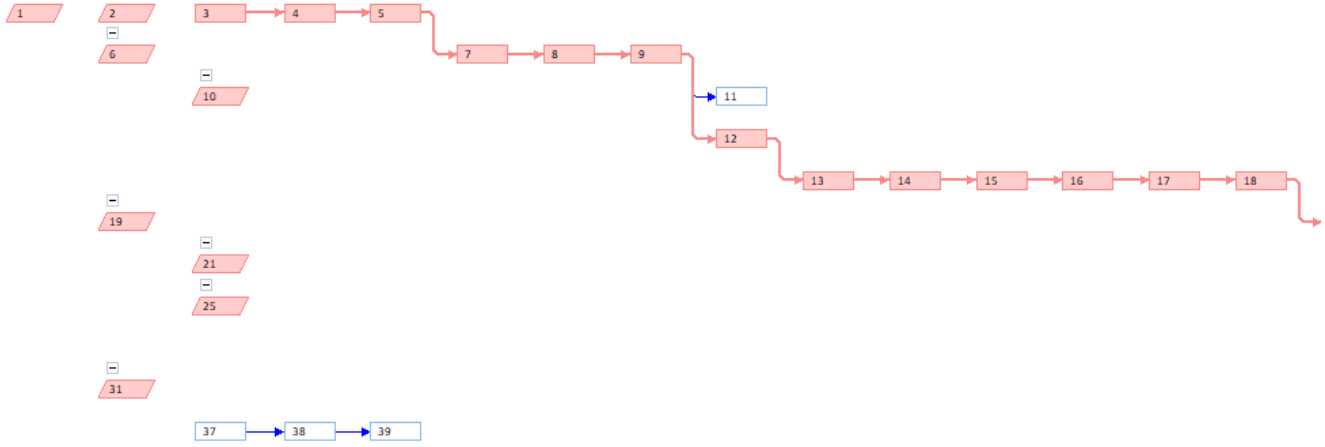

Рисунок А.4 - PDM мережа

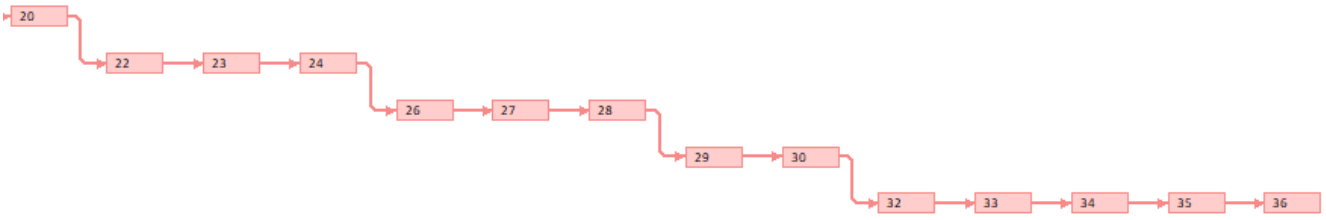

Рисунок А.5 - Продовження PDM мережі

Побудова календарного графіку

Календарне планування включає процес створення розкладу, що враховує різні матеріально-технічні та трудові ресурси, з метою визначення строків виконання проекту. Розклад містить інформацію про запланований та фактичний час початку, тривалість та завершення проекту, використовуючи WBS як основу для розподілу робіт.

Календарний графік даного проекту у вигляді діаграми Ганта зображено на рисунку А.6.

Хмарний додаток демонстрації та редагування дизайну приміщення з використанням VR технологій	**99 days**	**Mon 02.01.23**	**Thu 18.05.23**		
1. Ініціалізація проекту	**4 days**	**Mon 02.01.23**	**Thu 05.01.23**		Толстоноженко;Шендрик
1.1 Оцінка ідеї	1 day	Mon 02.01.23	Mon 02.01.23		Шендрик
1.2 Розробка концепції проекту	2 days	Tue 03.01.23	Wed 04.01.23	3	Толстоноженко;Шендрик
1.3 Затвердження ідеї проекту	1 day	Thu 05.01.23	Thu 05.01.23	4	Толстоноженко;Шендрик
2. Планування проекту	**13 days**	**Fri 06.01.23**	**Tue 24.01.23**		Толстоноженко;Шендрик
2.1 Визначення вимог	1 day	Fri 06.01.23	Fri 06.01.23	5	Толстоноженко;Шендрик
2.2 Визначення ПЗ	2 days	Mon 09.01.23	Tue 10.01.23	7	Толстоноженко
2.3 Визначення типу хостингу	1 day	Wed 11.01.23	Wed 11.01.23	8	Толстоноженко
2.4 Визначення системи керування базами даних	**1 day**	**Thu 12.01.23**	**Thu 12.01.23**		Толстоноженко
2.4.1 Для збереження даних файлів	1 day	Thu 12.01.23	Thu 12.01.23	9	Толстоноженко
2.4.2 Для збереження даних віртуальної кімнати та користувачів	1 day	Thu 12.01.23	Thu 12.01.23	9	Толстоноженко
2.5 Побудова WBS	1 day	Fri 13.01.23	Fri 13.01.23	12	Толстоноженко
2.6 Побудова OBS	1 day	Mon 16.01.23	Mon 16.01.23	13	Толстоноженко
2.7 Побудова матриці відповідальності	1 day	Tue 17.01.23	Tue 17.01.23	14	Толстоноженко
2.8 Оптимізація плану проекту	2 days	Wed 18.01.23	Thu 19.01.23	15	Толстоноженко;Шендрик
2.9 Створення мережевого графіка	2 days	Fri 20.01.23	Mon 23.01.23	16	Толстоноженко;Шендрик
2.10 Визначення ризиків проекту	1 day	Tue 24.01.23	Tue 24.01.23	17	Толстоноженко;Шендрик
3. Реалізація проекту	**66 days**	**Wed 25.01.23**	**Wed 26.04.23**		Толстоноженко
3.1 Установка ПЗ	2 days	Wed 25.01.23	Thu 26.01.23	18	Толстоноженко
3.2 Створення веб-сервера	**25 days**	**Fri 27.01.23**	**Thu 02.03.23**		Толстоноженко
3.2.1 Моделювання структури сервера	2 days	Fri 27.01.23	Mon 30.01.23	20	Толстоноженко
3.2.2 Розробка сервера	20 days	Tue 31.01.23	Mon 27.02.23	22	Толстоноженко
3.2.3 Тестування сервера	3 days	Tue 28.02.23	Thu 02.03.23	23	Толстоноженко
3.3 Створення плагіну для Unreal Engine 4	**25 days**	**Fri 03.03.23**	**Thu 06.04.23**		Толстоноженко
3.3.1 Моделювання структури плагіна	2 days	Fri 03.03.23	Mon 06.03.23	24	Толстоноженко
3.3.2 Розробка плагіна	20 days	Tue 07.03.23	Mon 03.04.23	26	Толстоноженко
3.3.3 Тестування плагіна	3 days	Tue 04.04.23	Thu 06.04.23	27	Толстоноженко
3.4 Тестування функції підключення плагіну до сервера	10 days	Fri 07.04.23	Thu 20.04.23	28	Толстоноженко
3.5 Загальне тестування	4 days	Fri 21.04.23	Wed 26.04.23	29	Толстоноженко
4. Завершення проекту	**16 days**	**Thu 27.04.23**	**Thu 18.05.23**		Толстоноженко;Шендрик
4.1 Перенос проекта на хостинг	2 days	Thu 27.04.23	Fri 28.04.23	30	Толстоноженко
4.2 Тестування на хостингу	2 days	Mon 01.05.23	Tue 02.05.23	32	Толстоноженко
4.3 Розробка документації	4 days	Wed 03.05.23	Mon 08.05.23	33	Толстоноженко;Шендрик
4.4 Відкривання загального доступу	1 day	Tue 09.05.23	Tue 09.05.23	34	Толстоноженко
4.5 Перевірка на плагіат	7 days	Wed 10.05.23	Thu 18.05.23	35	Шендрик
4.6 Рецензування	3 days	Tue 02.05.23	Thu 04.05.23		Толстоноженко;Шендрик
4.7 Перевірка керівником	2 days	Fri 05.05.23	Mon 08.05.23	37	Шендрик
4.8 Усунення недоліків	3 days	Wed 10.05.23	Fri 12.05.23	38	Толстоноженко

Рисунок А.6 - Діаграма Ганта проєкту

Управління ризиками проекту

Під час реалізації проекту можуть виникнути небажані умови, ситуації та наслідки, які називаються ризиками. Управління ризиком полягає у відповіді на ці події та зміни ризиків під час виконання проекту шляхом моніторингу та контролю за ними.

Для врахування ймовірності появи ризикових подій, які можуть мати негативні наслідки, можна створити таблицю класифікації ризиків. Потім на основі цієї таблиці створюється матриця ризиків.

Класифікація ризиків проекту наведено на таблиці А.4 а матриця ризиків наведено рисунок А.7.

Таблиця А.4 - Класифікація ризиків

№	Назва ризику	Ймовірність	Величина втрат
1	Некоректне складання технічного завдання	2	4
2	Порушення термінів виконання	4	4
3	Помилки у роботі веб-сервера	2	2
4	Несправності у пристроях	2	3
5	Недостатнє тестування	4	3
6	Втрата чи пошкодження інформації	2	5

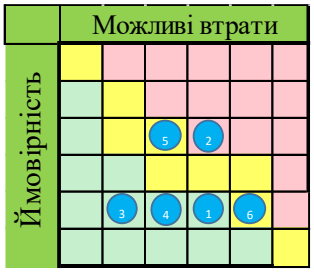

Рисунок А.7 – Матриця ризиків