Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Навчально-науковий
інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
з кваліфікаційної роботи ступеня бакалавра

здобувача Телятника Віталія Миколайовича
(ПІБ)

академічної групи 123-21-1
(шифр)

спеціальності 123 Комп'ютерна інженерія
(код і назва спеціальності)

за освітньою програмою Ком'ютерна інженерія
(офіційна назва)

на тему "Програмно-апаратний комплекс компанії з реалізацією функції навчання здобувачів технологіям корпоративних мереж"
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтингова	**інституційна**	
Кваліфікаційної роботи	проф. Цвіркун Л.І			
Спеціальної частини	проф. Цвіркун Л.І			

Рецензент				

Нормоконтролер	проф. Цвіркун Л.І			

Дніпро
2025

ЗАТВЕРДЖЕНО:
завідувач кафедри
інформаційних технологій
та комп'ютерної інженерії
(повна назва)
_____ Гнатушенко В.В.
(підпис) (прізвище, ініціали)
"____" *червня 2025 року*

ЗАВДАННЯ
на кваліфікаційну роботу ступеня бакалавра

здобувача Телятника В.М. **академічної групи** 123-21-1
(прізвище та ініціали) (шифр)
спеціальності 123 Ком'ютерна інженерія
за освітньо-професійною програмою Ком'ютерна інженерія
(офіційна назва)
на тему "Програмно-апаратний комплекс компанії з реалізацією функції навчання здобувачів технологіям корпоративних мереж" затверджену наказом ректора НТУ "Дніпровська Політехніка" від 05.05.2025 №336-с

Розділ	Зміст	Термін виконання
Стан питання і постановка задачі	На основі матеріалів виробничих практик та існуючих рішень обґрунтовано актуальність, сформульовано мету, задачі та загальну постановку проблеми.	05.05.2025
Розробка апаратної частини	Визначено призначення та технічні вимоги до системи, архітектури, функціональну структуру та навчальну логіку.	20.05.2025
Розробка програмного забезпечення комплексу	Обґрунтовано вибір технологій, реалізовано інтерфейс користувача, навчальні сцени та логіку взаємодії. Виконано експорт і публікацію застосунку.	25.05.2025
Тестування і оцінка ефективності	Проведено функціональне тестування, оцінено зручність користування, ефективність навчання та проаналізовано зворотний зв'язок.	30.05.2025

Завдання видано _____ проф. Цвіркун Л.І.
Дата видачі 25.02.2025 (підпис керівника)
Дата подання до екзаменаційної комісії 16.06.2025
Прийнято до виконання _____ Телятник В.М.

РЕФЕРАТ

Пояснювальна записка: 79 с., 16 рис., 1 табл., 1 дод., 8 джерел

КОМП'ЮТЕРНА МЕРЕЖА, IP-АДРЕСАЦІЯ, UNITY, ГЕЙМІФІКАЦІЯ, WEBGL, PACKET TRACER, ТОПОЛОГІЯ МЕРЕЖІ, НАВЧАЛЬНИЙ ЗАСТОСУНОК, МЕРЕЖЕВЕ МОДЕЛЮВАННЯ, ІНТЕРАКТИВНЕ НАВЧАННЯ.

Об'єкт дослідження – процес навчання основам комп'ютерних мереж за допомогою інтерактивних технологій.

Мета роботи – створення ефективного програмно-апаратного комплексу, який забезпечує можливість вивчення основ комп'ютерних мереж через гейміфікований WebGL-застосунок.

У роботі реалізовано програмний застосунок на базі рушія Unity, що імітує налаштування локальної мережі у форматі симулятора. Застосунок дозволяє користувачу додавати пристрої, з'єднувати їх, налаштовувати IP-адреси, DHCP, маршрути та перевіряти зв'язність за допомогою інтерактивного пінгу.

Контроль виконання дій здійснюється в реальному часі через систему логічних перевірок, а результати фіксуються локально у вигляді JSON-збереження. Процес навчання організовано у форматі пісочниці без жорсткої послідовності дій, що стимулює експериментальний підхід.

Функціональність протестовано у браузерному середовищі, а логіка моделювання схожа на Cisco Packet Tracer. Скріншоти з основними прикладами роботи наведені у додатку.

Практичне значення роботи полягає в можливості використання створеного комплексу в закладах освіти, для майбутніх IT-фахівців.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАК, ОДИНИЦЬ І ТЕРМІНІВ

IP Internet Protocol – інтернет-протокол

HTTP HyperText Transfer Protocol – протокол передавання гіпертексту

DHCP Dynamic

Host Configuration Protocol – протокол динамічного конфігурування хостів

DNS Domain Name System – система доменних імен

NAT Network Address Translation – трансляція мережевих адрес

VLAN Virtual Local Area Network – віртуальна локальна мережа

ACL Access Control List – список керування доступом

CLI Command Line Interface – інтерфейс командного рядка

PC Personal Computer – персональний комп’ютер

UI User Interface – інтерфейс користувача

UX User Experience – досвід користувача

FPS Frames Per Second – кількість кадрів за секунду

TCP/IP Transmission Control Protocol / Internet Protocol – стек мережевих протоколів

LAN Local Area Network – локальна мережа

WAN Wide Area Network – глобальна мережа

Wi-Fi Wireless Fidelity – бездротова передача даних

OSI Open Systems Interconnection – еталонна модель взаємодії відкритих систем

Unity Unity Game Engine – середовище розробки для створення ігор та інтерактивних додатків

WebGL Web Graphics Library – бібліотека для відмальовки 2D та 3D графіки у браузері

JSON JavaScript Object Notation – формат обміну даними

CI/CD Continuous Integration / Continuous Deployment – безперервна інтеграція / розгортання

CRM Customer Relationship Management – система керування взаємовідносинами з клієнтами

VPN Virtual Private Network – віртуальна приватна мережа

Packet Tracer Cisco Packet Tracer – середовище для моделювання комп’ютерних мереж

MVC Model-View-Controller – шаблон архітектури програмного забезпечення

Zenject – бібліотека для реалізації IoC контейнерів

UniTask - бібліотека для покращення асинхронного програмування в Unity

ReactiveProperty (R3) – бібліотека реактивних властивостей

LINQ Language Integrated Query – інтегровані запити до колекцій у C#

DOTween Demigiant Tweening Engine – бібліотека анімацій для Unity

ВСТУП

У сучасних умовах стрімкого розвитку цифрових технологій комп’ютерні мережі відіграють одну з ключових ролей у забезпеченні комунікації, обробки та збереження інформації. Ефективне використання комп'ютерних мереж сприяє розвитку підприємств, освітніх установ і громадських організацій. Зі зростанням попиту на кваліфікованих спеціалістів у сфері інформаційних технологій виникає потреба у вдосконаленні процесу навчання здобувачів відповідним навичкам та знанням, що зумовлює актуальність створення нових інтерактивних засобів навчання.

Світові тенденції розвитку освіти акцентують увагу на інтеграції гейміфікованих технологій, які сприяють кращому засвоєнню складного матеріалу за рахунок інтерактивності та залучення студентів. Гейміфікація навчального процесу дозволяє створити мотиваційне середовище, що стимулює здобувачів до активної участі у навчанні. Впровадження освітніх тренажерів і симуляцій на основі сучасних ігрових рушіїв, таких як Unity, стає ефективним інструментом для підготовки фахівців у сфері комп’ютерних мереж.

Особливої уваги потребує розробка спеціалізованих застосунків, орієнтованих на моделювання реальних мережевих процесів, що дозволяють користувачам не лише здобувати теоретичні знання, але й набувати практичних навичок роботи з мережевим обладнанням. Створення таких навчальних комплексів з урахуванням сучасних вимог до інформаційної безпеки є важливим кроком у підготовці майбутніх IT-спеціалістів.

Актуальність даної кваліфікаційної роботи полягає у розробці програмно-апаратного комплексу, що об'єднує можливості інтерактивного навчання, мережевого моделювання та самостійної перевірки знань здобувачів. Основна увага приділяється поясненню базових понять комп’ютерних мереж, таких як IP-адресація, підмережі, маршрутизатори та комутатори, з подальшим закріпленням знань через практичні завдання і тести.

Метою роботи є розробка програмно-апаратного комплексу компанії, що реалізує функцію навчання здобувачів технологіям комп'ютерних мереж шляхом створення гейміфікованого застосунку на базі рушія Unity у форматі WebGL. Застосунок включає навчальні модулі, тести та інтерактивні завдання, що дозволяють користувачам у простому 2D-середовищі опанувати налаштування мережевого обладнання, IP-адресацію та роботу з базовими мережевими топологіями.

Мережа спроєктована з урахуванням сучасних вимог до інформаційної безпеки, оптимізації адресного простору за допомогою IP-структурування, та базової маршрутизації. Для перевірки працездатності та безпеки запропонованої мережі використовуються моделювання і тестування в середовищі Cisco Packet Tracer.

Сфера застосування результатів роботи охоплює навчальні заклади різного рівня акредитації, спеціалізовані IT-курси, корпоративні програми підготовки фахівців, а також самостійне навчання осіб, що прагнуть розвивати навички в галузі комп’ютерних мереж. Створений програмно-апаратний комплекс забезпечує можливість як самостійного навчання, так і контролю прогресу за допомогою інтегрованих механізмів самоперевірки здобутих знань.

1 СТАН ПИТАННЯ І ПОСТАНОВКА ЗАВДАННЯ

1.1 Огляд сучасних методів навчання комп’ютерним мережам

Навчання комп’ютерним мережам є важливою складовою підготовки фахівців у сфері інформаційних технологій. Для якісного опанування цієї галузі недостатньо лише теоретичних знань – ключову роль відіграють практичні навички з налаштування, обслуговування та діагностики мережевих систем. Саме тому в освітньому процесі активно застосовуються спеціалізовані програмні засоби й симулятори, що дозволяють створювати й тестувати віртуальні мережі без потреби в реальному обладнанні.

Одним із найпоширеніших інструментів є Cisco Packet Tracer – симулятор, який дозволяє студентам віртуально створювати топології, налаштовувати маршрутизатори й комутатори, а також аналізувати роботу мереж. Завдяки простому у використанні інтерфейсу, наявності готових шаблонів пристроїв і широкій підтримці функцій, Packet Tracer використовується в багатьох навчальних закладах для вивчення таких тем, як IP-адресація, маршрутизація, VLAN, ACL, DHCP, NAT тощо.

Більш професійним рішенням є GNS3 (Graphical Network Simulator 3) – інструмент, що підтримує емуляцію реального мережевого ПЗ (Cisco IOS, JunOS) і дозволяє створювати складні мережеві конфігурації. Проте, його використання часто пов’язане з високими вимогами до комп’ютерних ресурсів і потребою в поглиблених знаннях мережевих технологій, що обмежує його придатність для новачків.

Крім симуляторів, в навчальному процесі застосовуються й онлайн-платформи, такі як Cisco Networking Academy, Google Labs, Microsoft Learn, Coursera, Udemy та інші. Вони пропонують систематизовані навчальні

курси з відеоматеріалами, тестами та інтерактивними вправами. Частина таких платформ інтегрує симулятори для практики, але часто розділяє теоретичний і практичний компоненти, що знижує ефективність засвоєння матеріалу.

Попри функціональну насиченість зазначених платформ, у більшості з них відсутній елемент безпосереднього занурення у навчання через інтуїтивно зрозумілу, інтерактивну та візуально орієнтовану взаємодію. Зокрема, студенту зазвичай потрібно адаптуватись до специфічного інтерфейсу, виконувати налаштування у командному рядку, мати окрему реєстрацію, а також витрачати час на вивчення інструкцій замість фокусування на самому матеріалі.

Сучасні дослідження в галузі освітніх технологій показують, що візуалізація навчального процесу та інтерактивність значно підвищують ефективність навчання. Наявність середовища, де користувач може експериментувати з конфігурацією пристроїв у зручному UI, бачити результат у реальному часі, отримувати підказки чи повідомлення про помилки — є важливою умовою для ефективного навчання. У цьому контексті все більшої актуальності набувають sandbox-підходи, де користувач має свободу дій у відкритому середовищі без нав’язаних обмежень сценаріїв.

На сьогодні більшість доступних рішень не реалізують подібної моделі. У Packet Tracer, хоча й доступна часткова візуалізація, взаємодія все ще обмежена й не завжди інтуїтивна. GNS3 орієнтований передусім на сертифікованих фахівців. Онлайн-платформи часто є фрагментарними, не підтримують інтеграції з інтерфейсом конфігурації обладнання.

У зв’язку з цим постає потреба у створенні нового освітнього інструменту, який би поєднував простоту та наочність, відкритість дій

користувача (sandbox), інтерактивні можливості симуляції, підказки й перевірки, а також можливість розміщення на відкритому хостингу для простого доступу без реєстрації. Такий підхід дозволяє сформувати сучасне навчальне середовище, що підвищує залучення здобувачів освіти, сприяє розвитку системного мислення й робить навчання гнучким і доступним.

Однією з технологій, що дозволяє реалізувати такий підхід, є середовище розробки Unity з можливістю експорту у формат WebGL, що дозволяє запускати симулятор без встановлення, просто у браузері. Це відкриває широкі перспективи для впровадження сучасних освітніх рішень на основі інтерактивної візуалізації та гнучкої логіки побудови навчального середовища.

1.2 Проблеми існуючих рішень

Попри наявність широкого спектру інструментів для вивчення комп’ютерних мереж, таких як Cisco Packet Tracer, GNS3, а також онлайн-платформ Cisco Networking Academy, Google Labs, Coursera та інших, їх використання супроводжується низкою проблем, що обмежують ефективність навчального процесу. Ці обмеження виявляються як на рівні подачі матеріалу, так і в організації взаємодії здобувача освіти з системою.

Одним із ключових недоліків є відсутність механізмів, що підвищують мотивацію користувача та створюють стале зацікавлення у навчанні. Більшість сучасних освітніх рішень орієнтовані на лінійне проходження навчальних модулів без використання гейміфікованих підходів. Студентам, особливо тим, хто навчається самостійно, бракує елементів залучення, таких як система заохочень, прогресивні завдання, візуальні підказки, інтерактивна допомога чи візуалізований результат виконання дій.

Іншим важливим аспектом є високий поріг входження для використання професійних симуляторів. Такі інструменти, як GNS3, вимагають глибоких технічних знань і потужного комп’ютерного обладнання, що унеможливлює їх використання на базовому освітньому рівні або у випадках обмежених ресурсів. Навіть менш вимогливі симулятори, як-от Cisco Packet Tracer, мають досить специфічний інтерфейс, який може бути незрозумілим або непривітним для студентів-початківців.

Окремою проблемою є відсутність повноцінної інтеграції між теоретичною та практичною частинами навчання. Часто студенту необхідно переходити між різними платформами для опрацювання теорії та практики, що ускладнює засвоєння матеріалу й руйнує цілісність навчального досвіду.

У більшості наявних рішень теоретичні курси не мають прямого зв’язку з практичними симуляціями, що знижує ефективність навчання.

До цього додається й проблема доступності ресурсів. Багато платформ вимагають обов’язкової реєстрації, постійного інтернет-з’єднання або мають обмеження у функціональності без придбання ліцензії. Це створює додаткові бар’єри для користувача, особливо в умовах самостійного чи неформального навчання.

Таким чином, хоча сучасні інструменти для вивчення комп’ютерних мереж демонструють значні технічні можливості, вони часто не відповідають вимогам сучасної педагогіки та очікуванням користувача щодо інтерактивності, гнучкості та доступності. Це формує актуальну потребу в розробці нового типу програмного забезпечення — інтерактивного веб застосунку з простим і зрозумілим інтерфейсом, що поєднує візуалізацію, симуляцію, підказки та елементи ігрової мотивації, забезпечуючи тим самим комплексний підхід до навчання.

1.3 Мета та завдання кваліфікаційної роботи

У сучасному освітньому просторі спостерігається стрімке зростання попиту на ефективні, доступні та гнучкі засоби навчання в галузі інформаційних технологій. Одним із ключових напрямів підготовки фахівців є вивчення основ комп'ютерних мереж – зокрема IP-адресації, налаштування мережевого обладнання, принципів маршрутизації та побудови локальної інфраструктури. Однак класичні методи викладання, засновані переважно на теоретичному підході або використанні складних симуляторів, втрачають актуальність у контексті потреб сучасного здобувача освіти.

Метою даної кваліфікаційної роботи є створення програмно-апаратного комплексу, що поєднує в собі інтерактивну освітню платформу, візуалізоване середовище для навчання та механізми самоперевірки. Комплекс реалізується у вигляді гейміфікованого WebGL-застосунку, розробленого на основі ігрового рушія Unity. Основною ціллю такого підходу є не лише забезпечення доступу до навчального контенту, але й підвищення залученості користувачів, мотивації до самостійного опрацювання матеріалу, а також закріплення практичних навичок через моделювання та взаємодію.

Запропонований продукт не передбачає фізичного налаштування мережевого обладнання, проте дозволяє користувачам симулювати типовий цикл взаємодії в умовному мережевому середовищі. Користувачі в ігровому форматі виконують послідовність дій, що відповідають базовим практикам: налаштування IP-адрес, підключення пристроїв, усунення логічних помилок конфігурації тощо. Таким чином, досягається ефект занурення у контекст професійної діяльності без ризику пошкодження реального обладнання та з можливістю повторення сценаріїв скільки завгодно разів.

У межах реалізації мети визначено низку завдань, що мають бути вирішені в межах цієї роботи:

– здійснити огляд існуючих підходів до навчання комп’ютерним мережам, визначити їх сильні та слабкі сторони, зокрема в аспекті доступності, візуалізації, гнучкості та гейміфікації;

– сформувати концепцію нового навчального засобу, яка базуватиметься на доступності через браузер, кросплатформності, інтуїтивному інтерфейсі та вбудованій логіці перевірки знань;

– спроєктувати архітектуру програмного комплексу з розділенням на функціональні модулі: навчальний контент, логіку ігрового процесу, інтерфейс користувача та мережеву публікацію;

– реалізувати програмну частину застосунку за допомогою середовища Unity, створити навчальний рівень, що симулює взаємодію з віртуальними ПК, маршрутизаторами та іншими умовними мережевими елементами;

– забезпечити можливість розміщення застосунку на вебплатформі (зокрема itch.io) для організації доступу без додаткових інсталяцій, з перспективою масштабування у форматі масового онлайн-навчання;

– провести тестування застосунку, оцінити його ефективність з точки зору функціональності, зрозумілості, зручності використання та дидактичного впливу на користувачів.

Реалізація цих завдань дозволить створити програмно-апаратний продукт, який відповідає сучасним вимогам до цифрової освіти, поєднуючи технічну точність із гейміфікованим підходом, що покликаний підвищити інтерес до теми та сформувати стійкі знання у сфері комп’ютерних мереж.

1.4 Обґрунтування вибору інструментів

Вибір інструментів розробки програмно-апаратного комплексу базується на необхідності забезпечити низку ключових вимог, які є критичними для реалізації освітнього застосунку сучасного зразка. Серед основних вимог можна виділити: інтерактивність, наочність, доступність, кросплатформність, підтримку гейміфікації, можливість швидкого розгортання та масштабування проєкту.

З урахуванням цих критеріїв, платформою для розробки було обрано середовище для ігрової розробки Unity, яке надає широкі функціональні можливості для створення інтерактивних середовищ, симуляцій, логіки сценаріїв, візуалізації навчального контенту, а також зручного управління ігровими об’єктами. Unity є популярною кросплатформною технологією з великою спільнотою розробників, широкою документацією та стабільною підтримкою, що робить його оптимальним інструментом як для ігрової індустрії, так і для освітніх симуляторів.

Ключовою перевагою Unity є підтримка експорту у формат WebGL – технології, яка дозволяє запускати застосунки безпосередньо в браузері, без необхідності інсталяції додаткового програмного забезпечення або плагінів. Такий підхід забезпечує максимально простий доступ для користувачів, незалежно від операційної системи або пристрою. В умовах зростаючої популярності дистанційного та мобільного навчання, ця властивість є надзвичайно важливою.

Для реалізації логіки застосунку, а також контролю поведінки об’єктів у віртуальному середовищі, використовується мова програмування C#, що є основною для Unity. C# дозволяє будувати чітку структуру коду, ефективно управляти подієвими сценаріями, реалізовувати системи зберігання прогресу,

обробки відповідей, генерації навчальних ситуацій та гейміфікаційних елементів – зокрема балів, досягнень, підказок тощо.

Застосунок розгортається на платформі itch.io, що виступає у якості безкоштовного хостингу для WebGL-проєктів. Платформа надає зручний інтерфейс для публікації, кастомізації сторінки проєкту, додавання документації, а також інтеграції статистики переглядів та запусків.

Візуальні елементи, які імітують мережеві пристрої (ПК, комутатори, маршрутизатори), створюються у вигляді 2D-об’єктів та інтерактивних компонентів, що дозволяють реалізувати логіку підключення, вибору IP-адреси, перевірки доступності тощо. Такий підхід дозволяє візуалізувати імітацію типових дій мережевого фахівця без використання фізичних пристроїв або складних емуляторів.

Таким чином, вибір технологій та інструментів є повністю обґрунтованим та спрямований на досягнення мети кваліфікаційної роботи – створення сучасного навчального засобу з високим рівнем інтерактивності, доступності та ефективності засвоєння знань.

1.5 Загальна постановка задачі

Ефективна підготовка фахівців у галузі комп’ютерних мереж потребує не лише вивчення теоретичного матеріалу, а й активного засвоєння практичних навичок через моделювання реальних ситуацій, експерименти та самостійну роботу із мережевим обладнанням. Один із сучасних підходів до реалізації такого навчального процесу полягає у створенні інтерактивного навчального середовища, яке дозволяє здобувачам освіти діяти у симульованому просторі, що імітує логіку реальної мережевої інфраструктури.

У межах цієї кваліфікаційної роботи поставлено задачу розробити навчальний програмно-апаратний комплекс у вигляді веборієнтованого застосунку, що поєднує візуальну доступність, зручність використання та інтерактивність. Його мета – надати користувачам можливість у вільному режимі досліджувати принципи побудови комп’ютерної мережі, взаємодії між пристроями, а також виконувати базові налаштування без потреби у фізичному обладнанні.

Запропонований застосунок реалізований як навчальна "пісочниця" (sandbox), у якій користувач самостійно додає та конфігурує мережеві пристрої – персональні комп’ютери, комутатори, маршрутизатори – з метою моделювання типової локальної мережі. Передбачена можливість налаштування IP-адрес, маски підмережі, шлюзу за замовчуванням, увімкнення DHCP, а також створення маршрутизованих топологій. Додатково реалізовано інструменти перевірки працездатності мережі, зокрема через функцію віртуального пінгу.

Навчальний процес супроводжується підказками та візуальними повідомленнями, що дозволяє користувачу аналізувати результати власних

дій, виявляти помилки у конфігурації та виправляти їх у реальному часі. Хоча застосунок не поділений на фіксовані рівні складності, користувач має змогу самостійно створювати як найпростіші сценарії підключення пристроїв, так і складні мережеві конфігурації з кількома маршрутизаторами.

Застосунок працює у середовищі веббраузера, завдяки чому не вимагає встановлення додаткового програмного забезпечення. Це досягається через використання технології WebGL, що дозволяє створювати інтерактивні 3D-сцени з високою продуктивністю. Хостинг реалізовано на платформі itch.io, яка забезпечує публічний доступ до застосунку та дозволяє відслідковувати базову статистику взаємодії користувачів.

Очікується, що запропоноване рішення може бути ефективно використане в університетських курсах, як допоміжний інструмент для лабораторних занять або самостійної роботи. Також воно може стати частиною освітніх програм у закладах професійної освіти, внутрішніх корпоративних тренінгах, або використовуватися у неформальному навчанні.

Таким чином, задача кваліфікаційної роботи полягає у створенні вебзастосунку, що моделює базову логіку функціонування комп’ютерної мережі, орієнтований на візуальну інтерактивність, простоту доступу та практичну цінність у навчальному процесі.

2 РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ ПРОГРАМНО-АПАРАТНОГО КОМПЛЕКСУ

Етап проєктування є визначальним при створенні програмно-апаратного комплексу. Важливо досягти балансу між технічною ефективністю, зручністю користування, дидактичною цінністю та відповідністю сучасним принципам цифрової педагогіки. У цьому розділі розглядається концепція побудови застосунку, описується його загальна структура, визначаються основні компоненти та взаємозв’язки між ними, а також обґрунтовується вибір архітектурного підходу.

Розроблюваний застосунок є симулятором у форматі навчального середовища-пісочниці, що дозволяє моделювати прості мережеві конфігурації. Користувач може додавати пристрої, з’єднувати їх кабелями, налаштовувати мережеві параметри та тестувати з’єднання за допомогою візуалізації пінг-запитів. Формування навичок відбувається шляхом експериментів у вільному середовищі, що сприяє розвитку логічного мислення та інтуїтивного розуміння структури комп’ютерних мереж. Навчання у такому середовищі не обмежене фіксованими рівнями чи сценаріями, що дозволяє адаптувати підхід під потреби різних категорій користувачів.

Архітектура програмного комплексу включає декілька основних складових. Програмна частина відповідає за логіку взаємодії, обробку дій користувача, реалізацію внутрішніх процесів, таких як створення з’єднань, пінг-запити, перевірка налаштувань. Візуальна частина відповідає за зображення пристроїв на сцені, надає можливість переміщувати об’єкти, виводити підказки та інтерфейсні елементи. Окрема увага приділена

організації доступу до застосунку: завдяки використанню WebGL продукт запускається прямо в браузері без потреби інсталяції, а хостинг на itch.io забезпечує легкий публічний доступ і можливість збору аналітики.

Кожен функціональний блок проєктується таким чином, щоб забезпечити взаємодію між усіма складовими системи. Це дозволяє забезпечити цілісну логіку, стабільну роботу та гнучкість у майбутньому розширенні функціоналу.

2.1 Технічні вимоги до комплексу

Апаратна частина програмно-апаратного комплексу призначена для забезпечення роботи освітнього вебзастосунку, що розміщується на сервері компанії з постійним віддаленим доступом через Інтернет. Комплекс використовується у функціонуючій компанії і є складовою інфраструктури, яка дозволяє користувачам з будь-якої точки світу взаємодіяти з навчальним середовищем через браузер.

Основу апаратного комплексу становлять сервер, маршрутизатор, комутатор і клієнтські персональні комп’ютери. Сервер виконує роль хосту для вебсторінки з інтерактивним застосунком, тоді як мережеве обладнання забезпечує маршрутизацію трафіку, роздачу IP-адрес через DHCP, вихід в Інтернет через NAT, а також локальну взаємодію між пристроями у межах офісної мережі.

Інформаційний обмін між компонентами комплексу здійснюється за допомогою стандартних мережевих протоколів, зокрема HTTP, TCP/IP та DHCP. Передбачено постійну наявність виходу до глобальної мережі через провайдера, що забезпечує стабільну доступність ресурсу для зовнішніх користувачів.

Комплекс має працювати у віддаленому режимі безперервно, з можливістю доступу до навчального контенту цілодобово. Фізичне розміщення передбачається в умовах звичайного офісного приміщення без особливих вимог до мікроклімату або безпеки.

Передбачається подальше масштабування інфраструктури до 50 робочих місць без суттєвих змін у мережевій топології. Така гнучкість дозволяє розширювати доступ до освітнього контенту та забезпечувати розвиток системи без перебудови всієї архітектури.

2.1.1 Найменування і призначення

Програмно-апаратний комплекс призначений для організації освітнього процесу в умовах реального офісного середовища. Основною його функцією є забезпечення доступу до інтерактивного вебзастосунку, який дозволяє користувачам вивчати основи комп’ютерних мереж через симуляцію типових задач.

Комплекс впроваджується на базі компанії, де програмна частина хостується на внутрішньому сервері з постійним віддаленим доступом через Інтернет. Завдяки цьому здобувачі освіти, незалежно від місця перебування, можуть підключатися до системи через браузер і взаємодіяти з навчальним середовищем.

Таким чином, призначення комплексу полягає у створенні централізованої платформи для навчання комп’ютерним мережам на основі віртуальної взаємодії, що відображає типову інфраструктуру компанії з сервером, маршрутизатором, комутатором та клієнтськими ПК.

2.1.2 Вимоги до структури і функціонування комплексу

Структура програмно-апаратного комплексу повинна забезпечувати стабільну та безперебійну роботу навчального вебзастосунку в умовах реального офісного середовища з гарантованим віддаленим доступом через Інтернет. Комплекс включає сервер, маршрутизатор, комутатор та клієнтські персональні комп’ютери, які формують єдину локальну мережу. Сервер виконує функцію хостингу застосунку та обробки запитів користувачів, маршрутизатор відповідає за маршрутизацію трафіку, призначення IP-адрес і вихід до глобальної мережі, а комутатор забезпечує зв’язок між усіма пристроями в межах локальної інфраструктури.

Мережевий обмін даними в межах комплексу здійснюється з використанням стандартних протоколів, таких як TCP/IP, HTTP і DHCP. Сервер повинен мати статичну IP-адресу та бути доступним ззовні за допомогою механізмів трансляції адрес на маршрутизаторі (NAT). Важливою вимогою є підтримка цілодобового функціонування системи без необхідності ручного втручання, з автоматичним відновленням у разі перезапуску.

Комплекс має бути здатним працювати у постійному віддаленому режимі з одночасною підтримкою взаємодії як з локальними, так і з зовнішніми користувачами. Передбачена можливість подальшого розширення локальної мережі до 50 робочих місць без необхідності повної модернізації інфраструктури, що забезпечує гнучкість і масштабованість рішення.

Таким чином, вимоги до структури та функціонування комплексу охоплюють як апаратне компонування, так і логіку взаємодії між елементами системи, з урахуванням умов реального використання в межах невеликої компанії з відкритим зовнішнім доступом до навчального сервісу.

2.1.3 Вимоги до показників призначення

Програмно-апаратний комплекс призначений для експлуатації в умовах офісного середовища з типовими параметрами мікроклімату, без потреби у спеціалізованих умовах або додаткових засобах захисту. Робота комплексу передбачена у звичайному приміщенні з доступом до електромережі, стабільним інтернет-з’єднанням та базовими умовами фізичної безпеки.

Основні показники призначення комплексу полягають у забезпеченні постійного віддаленого доступу до освітнього вебзастосунку для необмеженої кількості користувачів, із врахуванням можливості одночасної взаємодії з ресурсом із різних географічних локацій. Комплекс повинен відповідати вимогам щодо стійкої роботи серверної частини під навантаженням та забезпечення безперервної доступності ресурсу без переривання сеансу взаємодії.

У рамках масштабованості проєкту передбачається розширення інфраструктури до 50 клієнтських робочих місць. Це вимагає підтримки відповідної пропускної здатності мережі, стабільності маршрутизації та ефективного управління IP-адресним простором.

Призначення комплексу також передбачає можливість використання його як внутрішнього освітнього інструменту компанії або як публічного навчального сервісу, що не вимагає фізичного втручання з боку користувача для встановлення або налаштування – доступ до навчального середовища здійснюється виключно через веббраузер.

Таким чином, сукупність показників призначення охоплює здатність комплексу функціонувати в реальних умовах офісного середовища, відповідати запитам щодо доступності, стійкості та масштабованості, а також забезпечувати зручність використання в навчальних цілях.

2.2 Апаратне забезпечення комплексу

Апаратна складова програмно-апаратного комплексу виконує критичну роль у забезпеченні функціонування програмного середовища та підтримці стабільного доступу до навчального контенту. До складу комплексу входять сервер, маршрутизатор, комутатор та клієнтські персональні комп’ютери, що формують єдину логічну та фізичну мережу в межах офісного середовища.

Сервер є центральною ланкою системи, на якій розгорнуто вебзастосунок. Його обчислювальні ресурси повинні бути достатніми для обробки багатопотокових запитів користувачів, підтримки навчального середовища, а також збереження проміжних і кінцевих результатів взаємодії. Вимоги до серверного обладнання включають наявність сучасного багатоядерного процесора, не менш як 8 ГБ оперативної пам’яті, високошвидкісного накопичувача з резервуванням даних, а також підтримку сучасних вебтехнологій, зокрема WebGL, для коректної візуалізації контенту.

Маршрутизатор забезпечує вихід комплексу до глобальної мережі Інтернет, реалізує функції трансляції адрес (NAT), а також може виконувати роль DHCP-сервера. Він також повинен підтримувати механізми переадресації портів для забезпечення зовнішнього доступу до внутрішнього сервера. Комутатор відповідає за фізичне з’єднання пристроїв у локальну мережу та повинен підтримувати відповідну пропускну здатність з урахуванням перспективи розширення до 50 робочих місць.

Клієнтські персональні комп’ютери, як внутрішні, так і зовнішні, використовуються для взаємодії з навчальним застосунком через браузер. Для цього достатньо базового рівня технічних характеристик, які забезпечують сумісність із сучасними вебтехнологіями.

Загалом апаратна інфраструктура має бути оптимізована для забезпечення високої доступності, масштабованості та стійкості системи, при цьому зберігаючи відносну простоту реалізації в умовах обмеженого офісного простору.

2.2.1 Розробка структурної схеми комплексу та мережі

Для забезпечення ефективного функціонування програмно-апаратного комплексу розроблено структурну схему, що відображає всі ключові компоненти системи та їхню взаємодію. Побудова мережі базується на використанні окремого HTTP-сервера, локальної офісної мережі, корпоративного маршрутизатора з NAT та прямим підключенням до провайдера. Така архітектура забезпечує як внутрішню взаємодію пристроїв компанії, так і зовнішній доступ до навчального застосунку через Інтернет.

Згідно з таблицею 2.1, кожному сегменту мережі присвоєно власний діапазон IP-адрес. HTTP-сервер, на якому розміщено WebGL-застосунок, має статичну IP-адресу та доступний ззовні за допомогою механізму трансляції адрес на корпоративному маршрутизаторі. Робочі станції отримують IP-адреси динамічно через DHCP. Корпоративний маршрутизатор здійснює розподіл трафіку, виконує функції NAT, а також забезпечує з’єднання з маршрутизатором провайдера, який виступає межею між локальною та глобальною мережею.

Структурну схему взаємодії компонентів комплексу зображено на рисунку 2.1. Вона демонструє логічну побудову комплексу, зокрема, розміщення сервера всередині компанії та принцип організації доступу до нього через проксі-сервер itch.io. Окремо на рисунку 2.2 представлено мережеву топологію, яка ілюструє IP-структурування, послідовність маршрутизації та схему доступу з боку зовнішнього користувача.

Таблиця 2.1 – Структура локальної мережі програмно-апаратного комплексу

№	Сегмент	IP-адреса / діапазон	Призначення	Примітки
1	HTTP-сервер	192.168.0.2	Видає WebGL-застосунок	Статична IP, доступ ззовні через NAT
2	Корпоративні ПК	192.168.0.100–192.168.0.200	Робочі станції всередині компанії	DHCP
3	Корпоративний маршрутизатор	192.168.0.254/24 - LAN 10.0.0.1/30 - WAN	Забезпечує вихід в Інтернет та NAT	NAT-переадресація до HTTP-сервера
4	Маршрутизатор провайдера	10.0.0.2/30 - WAN	З’єднання з корпоративним маршрутизатором	Межа між локальною і глобальною мережею
5	Зовнішній користувач (Інтернет)	100.100.100.25/24	Доступ до сервера через itch.io	Через проксі itch.io + публічний порт

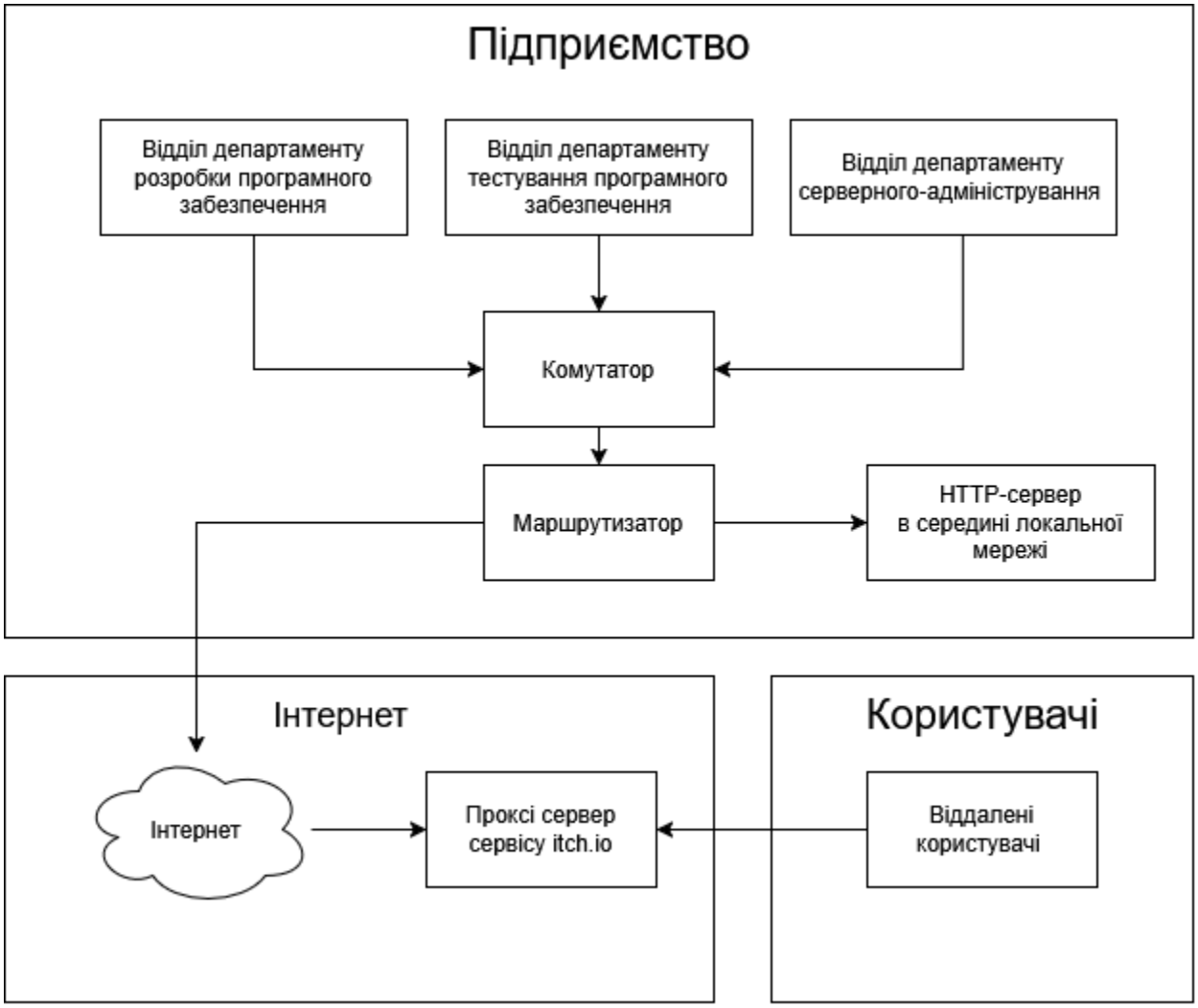

Рисунок 2.1 – Структурна діаграма архітектури програмного комплексу та взаємодії з користувачем

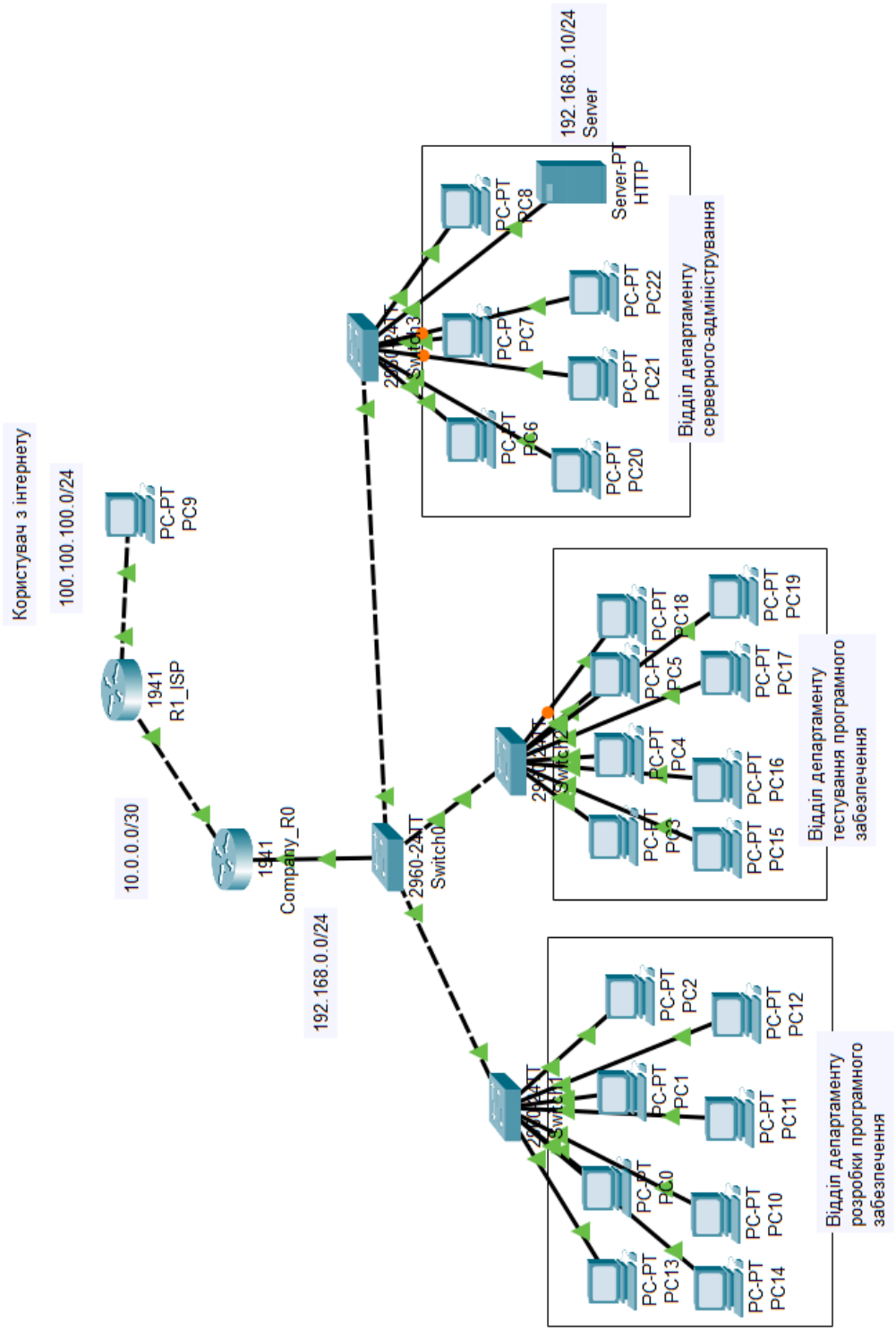

Рисунок 2.2 – Схема мережевої топології із зовнішнім HTTP-доступом

2.2.2 Розробка функціональної схеми комплексу

Функціональна схема програмно-апаратного комплексу відображає послідовність дій, що відбуваються під час взаємодії користувача з навчальним застосунком. Вона включає як мережеву частину, пов’язану з обробкою запиту до серверу, так і внутрішні програмні етапи, які забезпечують запуск, опрацювання даних та візуалізацію вмісту у браузері.

Процес починається із запиту користувача на отримання ігрових ресурсів. Цей запит передається через платформу itch.io або проксі-сервер, після чого надходить до внутрішнього HTTP-серверу підприємства, де розміщено WebGL-білд навчального середовища. Після успішної перевірки доступу, користувач отримує відповідь, що включає дозволи на завантаження ресурсу та ініціалізацію програми.

Після завантаження WebGL-програми в браузері запускається основний цикл програми. Кожен кадр відображення включає послідовні етапи: початок відмальовування, зчитування даних від користувача, опрацювання введених даних відповідно до сценарної логіки, завершення обробки кадру та оновлення зображення на екрані.

Уся взаємодія здійснюється в реальному часі, а цикл оновлення кадрів повторюється безперервно доти, доки користувач перебуває у навчальному середовищі. Такий підхід дозволяє забезпечити інтерактивність, миттєвий зворотний зв’язок і ефективну візуалізацію дій у контексті моделювання мережевих процесів.

Вказана логіка функціонування представлена на рисунку 2.3, де зображено взаємозв’язок між компонентами системи, мережевими запитами та внутрішньою структурою обробки даних у WebGL-програмі.

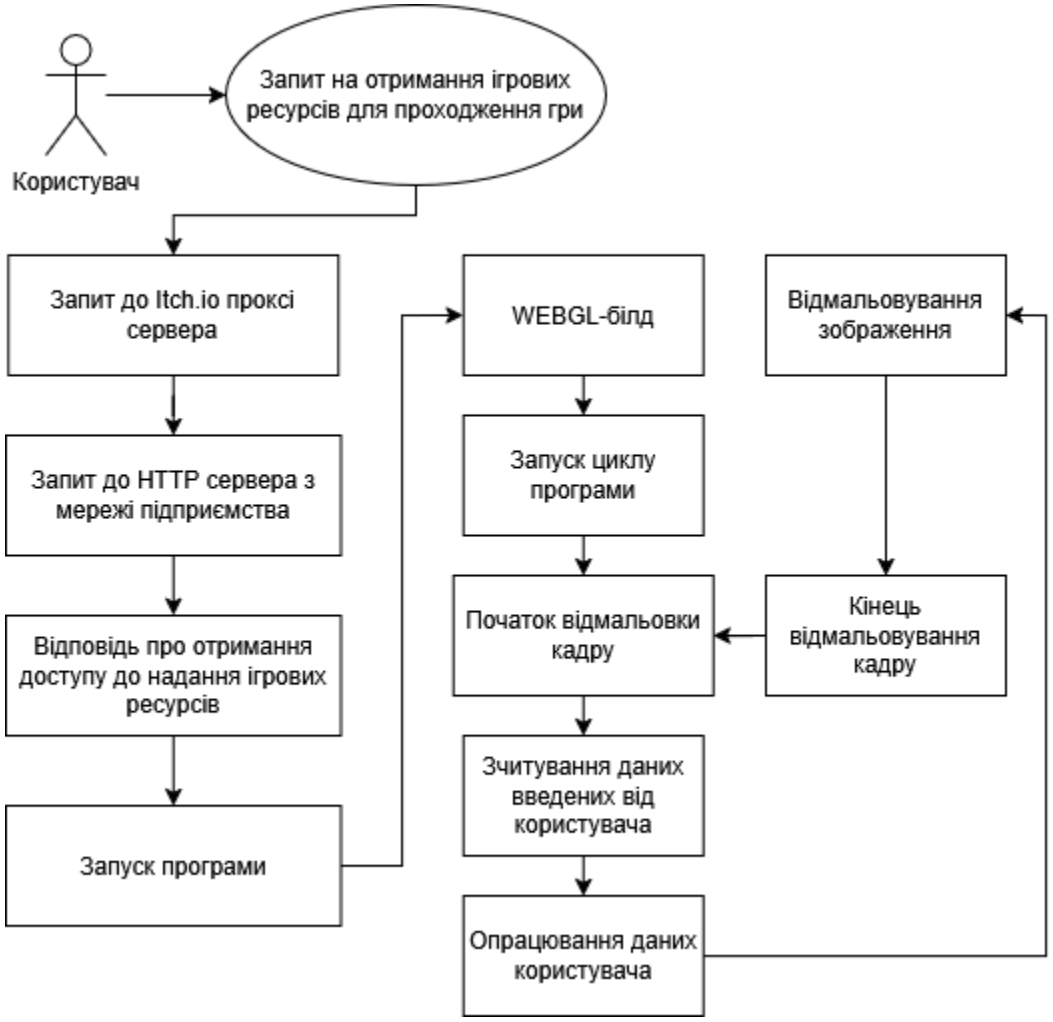

Рисунок 2.3 – Функціональна діаграма обробки запиту користувача у WebGL-програмі

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМПЛЕКСУ

Етап розробки застосунку реалізовано на основі рушія Unity, що забезпечив необхідну гнучкість, візуальну інтерактивність та кросплатформеність освітнього середовища. При створенні програмного продукту було дотримано принципів модульності, масштабованості та зручності підтримки, що дозволило ефективно організувати структуру коду, інтерфейсу та логіки взаємодії користувача з елементами сцени.

У процесі реалізації використовувались перевірені підходи до побудови ігрових застосунків, включаючи патерни проектування, інверсію керування через DI-фреймворк Zenject, реактивне програмування з використанням бібліотеки R3, асинхронну обробку дій на базі UniTask та візуальні ефекти з допомогою DOTween. Це дозволило зробити взаємодію користувача з навчальним середовищем динамічною, анімаційно плавною та логічно цілісною.

Інтерфейс користувача, побудований на UI-компонентах Unity, включає як базові елементи керування (меню, кнопки, текстові поля), так і спеціалізовані панелі, що забезпечують розміщення пристроїв, прокладання кабелів, керування параметрами ПК або маршрутизатора. Реалізована логіка інтерфейсу дозволяє інтуїтивно взаємодіяти з усіма ключовими функціями програми без потреби у додатковому навчанні або документації.

Розробка охоплює дві основні сцени: стартову, де користувач обирає дію або переглядає інструкції, та основну, в якій відбувається безпосередня взаємодія з навчальним контентом. Саме тут розміщується навчальне середовище, відбувається моделювання мережі та перевірка зв’язності.

Особливу увагу приділено реалізації взаємодії з мишкою: розміщення пристроїв, їх видалення, з’єднання кабелями, запуск пінгу – усе це відбувається безпосередньо у візуальному середовищі та відповідає діям, які знайомі користувачам з досвідом роботи у реальних системах моделювання мереж. Такий підхід дозволив досягти максимальної близькості симуляції до реального робочого середовища.

3.1 Призначення й сфера застосування програми

Розроблена програма є інтерактивним навчальним засобом, що реалізує моделювання процесів налаштування локальних комп'ютерних мереж у віртуальному середовищі. Основне функціональне призначення програми полягає у візуалізації дій, пов'язаних із конфігурацією мережевої інфраструктури, зокрема додаванням пристроїв, призначенням IP-адрес, увімкненням DHCP, налаштуванням маршрутів та перевіркою зв'язності між вузлами за допомогою команди ping. Застосунок дає змогу в інтерактивній формі формувати у користувача практичні навички, що відповідають базовому рівню підготовки фахівців у сфері комп'ютерних мереж.

Програма призначена для використання у процесі вивчення відповідних дисциплін у закладах вищої освіти, професійних навчальних центрах, на внутрішніх корпоративних тренінгах, а також для самостійного опанування матеріалу. Вона може бути використана як у межах традиційного навчального процесу, так і в дистанційному або змішаному форматі. WebGL-реалізація забезпечує широкі можливості доступу до ресурсу без інсталяції програмного забезпечення на комп'ютері користувача, що особливо актуально для впровадження в умовах обмежених технічних ресурсів.

Оскільки програмний продукт поєднує в собі елементи гейміфікації, візуальної взаємодії та самостійного конструювання мережевих конфігурацій, він дозволяє досягти високого рівня залучення здобувачів у навчальний процес. Завдяки гнучкій структурі сцени та відсутності жорстких сценарних обмежень, користувачі мають змогу експериментувати, перевіряти свої рішення та закріплювати знання через безпосередню взаємодію з навчальним середовищем. У результаті програма забезпечує ефективне поєднання теоретичного та практичного компонентів освітнього процесу.

3.2 Обґрунтування технічних характеристик

Для реалізації програмного комплексу було обрано середовище розробки Unity. Це середовище розробки стало оптимальним вибором завдяки своїм широким можливостям, активній спільноті та підтримці WebGL, що дозволило легко розгортати готовий застосунок без потреби у серверній інфраструктурі. Unity забезпечує зручну побудову ігрових сцен, обробку взаємодії користувача, а також підтримку компонентно-орієнтованої архітектури, що є надзвичайно корисним у контексті створення навчального середовища з інтерактивними об’єктами.

Основною мовою програмування, яка використовувалась у проєкті, є C#, що забезпечила ефективну роботу з логікою, подіями та візуальними об’єктами. Для організації архітектури проєкту було впроваджено фреймворк Zenject – рішення для реалізації принципу впровадження залежностей, яке дозволяє гнучко керувати життєвим циклом об’єктів і знижує ступінь зв’язаності між компонентами.

Для асинхронного виконання задач застосовувалась бібліотека UniTask, яка дозволила уникнути громіздкого використання IEnumerator та забезпечила легкість у читанні та підтримці коду. Реактивна логіка, що лежить в основі багатьох сценаріїв застосунку (зокрема, обробка змін у UI, стани мережевих пристроїв тощо), була реалізована за допомогою бібліотеки R3, що дозволила легко відстежувати та реагувати на зміни станів у застосунку.

Для реалізації плавної та інтуїтивної анімації було використано бібліотеку DOTween. З її допомогою реалізовано візуалізацію передачі пакетів під час пінгу, появу вікон і кнопок, а також інші візуальні ефекти, що покращують сприйняття взаємодії. Для обробки колекцій, пошуку потрібних

елементів, фільтрації та агрегації даних активно застосовувалась LINQ – стандартна частина мови C#.

Усі UI-компоненти були реалізовані з використанням стандартних інструментів Unity без залучення сторонніх UI-фреймворків. Це забезпечило стабільність та хорошу продуктивність при експорті у формат WebGL, а також просту адаптацію до різних розмірів екрану через систему якорів та базову адаптивність інтерфейсу.

Розробка програмного компонента навчального комплексу передбачала створення засобу для моделювання базових мережевих сценаріїв у віртуальному середовищі. Основною вимогою до технічної реалізації стало забезпечення стабільної роботи застосунку в браузері без необхідності встановлення додаткового програмного забезпечення.

Функціонування програми реалізовано в середовищі розробки Unity з експортом у формат WebGL. Такий підхід забезпечує сумісність із більшістю сучасних веб браузерів і не вимагає високих обчислювальних ресурсів. Усі дії користувача опрацьовуються через подієву модель, що дозволяє реагувати на взаємодії в реальному часі: додавання пристроїв, налаштування параметрів, перевірку зв’язності тощо.

Вхідні дані – це параметри, задані користувачем у графічному інтерфейсі, зокрема IP-адреси, режими роботи інтерфейсів, вибрані пристрої та типи з’єднань. Вихідними даними є візуалізовані результати (анімації пінгу, повідомлення про помилки або успішну конфігурацію) та JSON-файл збереженого стану сцени.

Вибір платформи розробки, формату обміну даними та інтерфейсної моделі забезпечив досягнення необхідної функціональності при збереженні доступності, гнучкості та можливості масштабування застосунку.

3.3 Опис розробленої програми

Розроблена програма є інтерактивним застосунком симуляторного типу, що реалізує навчальну пісочницю для моделювання локальних комп’ютерних мереж. Основу функціонування програми становить логіка створення та взаємодії між умовними мережевими пристроями, такими як персональні комп’ютери, комутатори та маршрутизатори. Кожен об’єкт у сцені має набір параметрів, які можуть бути змінені користувачем у процесі симуляції.

Функціональне призначення програми полягає у наданні користувачеві можливості візуального налаштування мережевих конфігурацій, введення та редагування IP-адрес, увімкнення або вимкнення DHCP, а також перевірки працездатності мережі за допомогою вбудованого інструменту пінгу. Усі дії реалізовані через інтерфейс, що забезпечує просту та наочну взаємодію без потреби використання командного рядка чи зовнішніх конфігураційних файлів.

Логічна структура програми побудована за модульним принципом. Основні підсистеми включають: менеджер об’єктів, систему валідації конфігурацій, візуальний інтерфейс користувача, а також контролери збереження та відновлення стану сцени. Програма підтримує механізм локального збереження конфігурації у форматі JSON, що дозволяє користувачу зафіксувати створену топологію та пізніше відновити її без втрати внесених змін.

Для роботи програми не потрібне спеціальне обладнання або потужні обчислювальні ресурси. Програма запускається безпосередньо у веб браузері, що робить його доступним для широкого кола користувачів.

3.3.1 Принципи побудови середовища

Навчальне середовище побудоване за принципами відкритості, гнучкості та варіативності. Основна ідея полягає у створенні умов для вільного експериментування з мережевими компонентами без фіксованого сценарію або послідовності дій. Користувач має змогу самостійно визначати цілі, розміщувати пристрої, з'єднувати їх, змінювати параметри та перевіряти взаємодію у реальному часі.

Сцена реалізована як єдина інтерактивна площина, на якій розміщуються всі доступні пристрої – комп’ютери, світчі, маршрутизатори. Створення об’єктів виконується вручну, з можливістю вибору типу пристрою через меню або панель. Для керування доступні базові дії: розміщення на полі, з’єднання кабелем, редагування параметрів, видалення. Усе це здійснюється інтуїтивно через мишу або елементи інтерфейсу.

При цьому важливою особливістю є повна свобода у виборі послідовності дій – користувач може спочатку налаштувати IP, потім підключити пристрої, або ж навпаки. Немає необхідності дотримуватися жорсткої структури, що дозволяє формувати індивідуальний підхід до навчання, спираючись на потреби конкретного здобувача.

Такий підхід базується на сучасних принципах проблемно-орієнтованого та дослідницького навчання, де ключову роль відіграє не заздалегідь визначений сценарій, а активна участь користувача у створенні та аналізі ситуацій.

3.3.2 Логіка та структура програмного забезпечення

Навчальне середовище, реалізоване у вигляді єдиної інтерактивної сцени, спроєктоване відповідно до сучасних педагогічних підходів, які ставлять у центр уваги практичний досвід, самостійність користувача та можливість вільного експериментування. У контексті моделювання комп'ютерних мереж це означає, що користувач отримує повну свободу дій: немає фіксованих рівнів, нав'язаного сценарію чи послідовності. Замість цього застосунок функціонує як умовна «пісочниця» (sandbox), де здобувач може в будь-який момент створити нову топологію, зруйнувати її, спробувати інший підхід або відтворити реальні мережеві ситуації.

Центральне місце в логіці взаємодії займає механіка створення та керування мережевими компонентами: комп'ютерами, світчами, маршрутизаторами та кабелями. Кожен пристрій має набір властивостей, які можна редагувати – наприклад, для комп'ютера це IP-адреса, маска підмережі, шлюз або режим DHCP. Для маршрутизатора доступна таблиця маршрутів і конфігурація інтерфейсів. Усі ці параметри змінюються у реальному часі, і результати дій миттєво відображаються у вигляді реакцій системи – чи то успішний пінг, чи відмова у зв'язку.

Важливою особливістю є відсутність жорсткої логіки сценаріїв: користувач може почати з простої конфігурації (наприклад, два ПК, з'єднані між собою) і поступово ускладнювати її, додаючи світчі, маршрутизатори, додаткові підмережі, змінюючи адресацію, активуючи DHCP або налаштовуючи статичні маршрути. Така гнучкість не лише стимулює дослідницький підхід, а й дозволяє адаптувати середовище під індивідуальні цілі навчання або конкретні приклади з реального життя.

Логіка перевірки дій реалізована у вигляді внутрішньої системи тригерів. Наприклад, при встановленні IP-адреси система перевіряє, чи відповідає вона масці підмережі; при запуску пінгу – чи існує правильний маршрут між пристроями; при використанні DHCP – чи активований відповідний сервер на іншому кінці мережі. Результати цих перевірок супроводжуються візуальним зворотним зв’язком: у разі успіху користувач бачить зелену анімацію між пристроями (рисунок 3.1), у разі помилки – червоне підсвічування та попередження.

Важливо також, що користувач має змогу будувати високорівневі сценарії з використанням двох або більше маршрутизаторів, які з’єднані між собою. Наприклад, конфігурація, де кожен маршрутизатор обслуговує свою підмережу, але з’єднані між собою через зовнішні інтерфейси. Це дозволяє вивчати маршрутизацію, роботу з таблицями маршрутів та міжмережеву зв’язність. Такі сценарії не тільки сприяють глибшому розумінню теми, але й дозволяють підготуватися до реальних викликів у професійній діяльності мережевого спеціаліста.

Для зручності вивчення, у застосунку реалізована можливість зберігати поточний стан сцени у вигляді JSON-структури, а згодом завантажувати його для повторного використання. Це дозволяє не лише продовжити роботу з перерваного місця, а й створювати власні шаблони навчальних конфігурацій. Таким чином, структура середовища виходить за межі статичної моделі, трансформуючись у динамічний простір для досліджень, перевірки гіпотез та закріплення знань.

У підсумку, така структура навчального середовища дозволяє говорити про переваги відкритої, інтерактивної моделі, де теоретичні знання миттєво перевіряються на практиці.

3.3.3 Структура задач і навчальних сцен

Навчальні задачі в симуляторі реалізуються не у вигляді заздалегідь прописаних рівнів, а як набір практичних дій, які користувач може виконувати всередині загальної сцени. Задачі носять умовний характер – це радше виклики, які користувач сам ставить перед собою в процесі дослідження функціональності мережі.

Наприклад, користувач може перевірити принцип роботи DHCP, створивши два комп’ютери, з’єднавши їх кабелем, активувавши опцію автоматичного отримання IP і спостерігаючи, чи було отримано адресу. Або ж він може побудувати мережу з декількох сегментів, з'єднаних через маршрутизатори, налаштувати таблицю маршрутів і перевірити зв’язність між віддаленими пристроями.

Виконання таких задач не обмежене. Користувач може перевіряти результат як через візуальний стан мережі, так і через спеціальні інструменти: команда ping, таблиця маршрутизації, панель параметрів пристроїв. Це забезпечує широкий спектр підходів до вирішення завдань – від простих перевірок IP до складних схем маршрутизації з кількома підмережами.

Сцена також підтримує збереження конфігурацій, що дозволяє зафіксувати певний стан як шаблон задачі або результат її вирішення. Таким чином, навчальні сцени є динамічним середовищем, де кожна дія користувача – потенційна навчальна ситуація.

3.3.3.1 Створення та налаштування навчальних сцен

Створення навчального середовища в Unity передбачало організацію двох основних сцен: стартової та основної ігрової. Стартова сцена виконує роль меню, де користувач може розпочати нову сесію, ознайомитися з інструкцією, переглянути доступні функції та налаштування. Основна ігрова сцена реалізує всю взаємодію користувача з мережевими пристроями, кабелями та механіками перевірки зв’язності.

Ігрове середовище побудовано на базі 2D-орієнтованої сцени, яка дозволяє розміщувати віртуальні пристрої за принципом вільного перетягування. Розміщення пристроїв відбувається у реальному часі – користувач обирає тип пристрою (ПК, світч або маршрутизатор), після чого об’єкт «слідкує» за курсором до моменту встановлення. Такий підхід створює ефект інтуїтивної побудови мережі без необхідності складних форм взаємодії (рисунок 3.2).

Кабелі створюються за допомогою LineRenderer і візуально з’єднують обрані пристрої. На першому натиску правої кнопки миші вибирається вихідний пристрій, після чого кабель тягнеться за курсором до моменту підтвердження другого пристрою. При цьому автоматично відображається список доступних портів, як показано на рисунку 3.4. Якщо порт зайнятий – його не буде в переліку.

У межах навчальної сцени реалізовано кілька типовий конфігурацій, які дозволяють продемонструвати принципи роботи різних мережевих протоколів. Наприклад, можна налаштувати локальну мережу з двох ПК, або ж складнішу – з маршрутизатором, світчем і підмережами, як видно на рисунках 3.3 та 3.5. Така варіативність дозволяє створювати завдання різного рівня складності.

Навчальна сцена також містить візуальні підказки, туторіал-вікно (рисунок 3.6), логіку активації режимів редагування та перевірки дій. Взаємодія з пристроями супроводжується окремими панелями, які дозволяють налаштувати IP-адресу, активувати DHCP (рисунок 3.7) або вручну вказати таблицю маршрутів (рисунок 3.5).

Таким чином, сцени реалізовано таким чином, щоби забезпечити максимальну гнучкість, інтуїтивність взаємодії та можливість легко масштабувати навчальний контент у майбутньому.

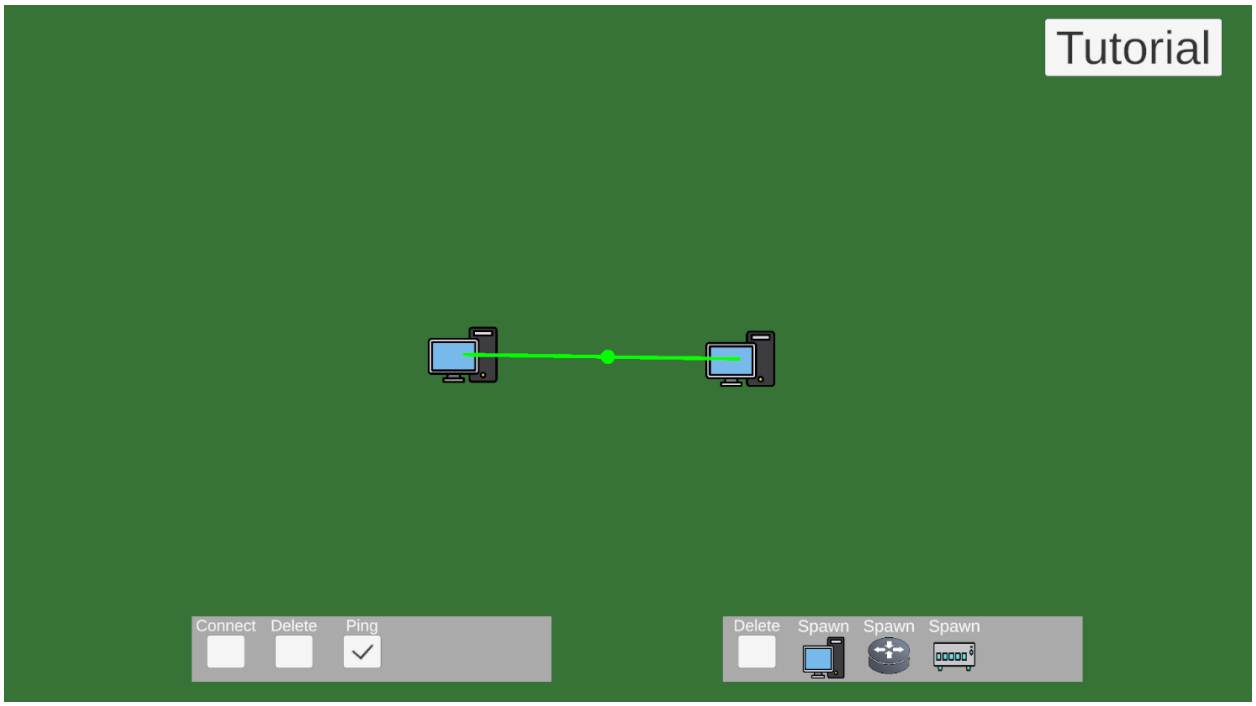

Рисунок 3.1 – Візуалізація успішного пінгу між ПК (зелений маркер)

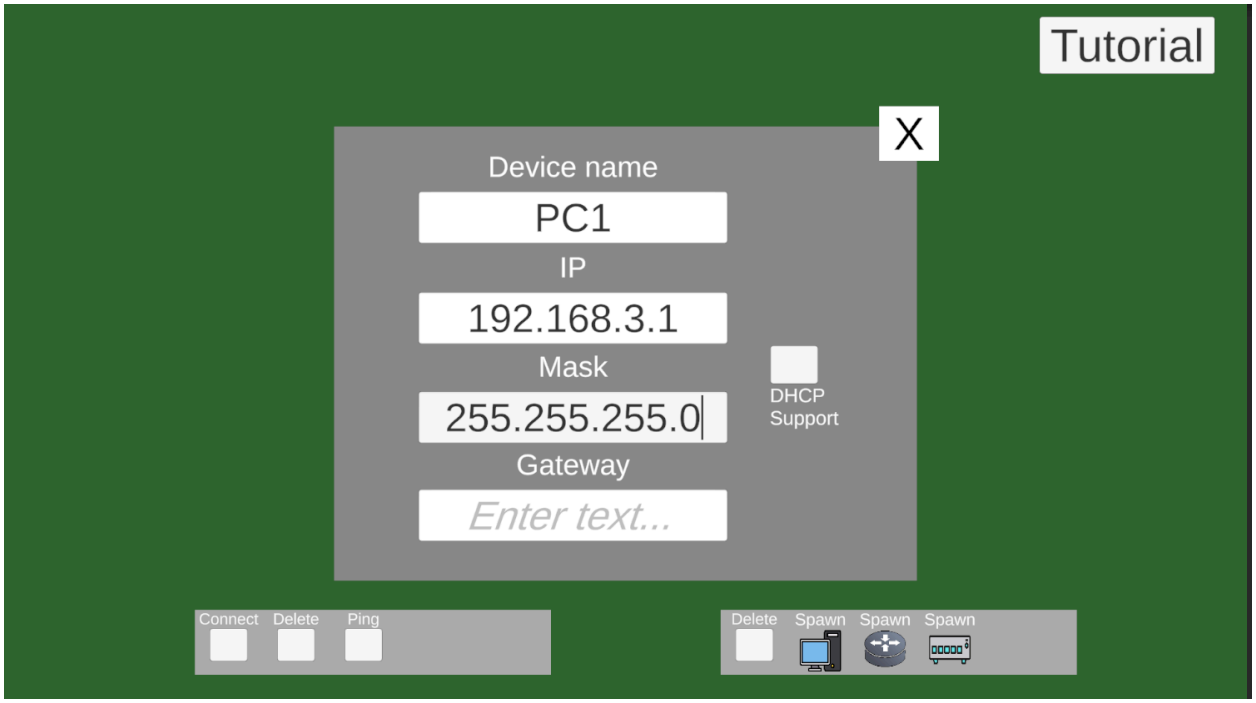

Рисунок 3.2 – Вікно налаштування параметрів комп’ютера

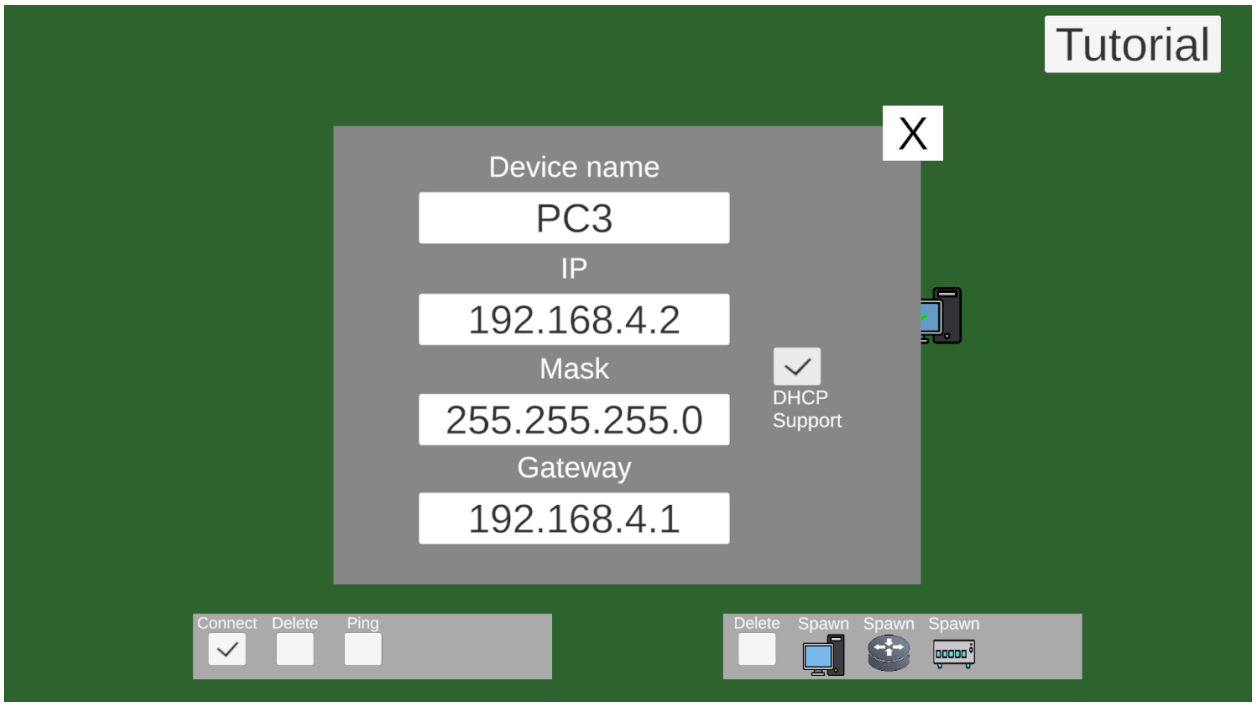

Рисунок 3.3 – Альтернативне налаштування іншого ПК

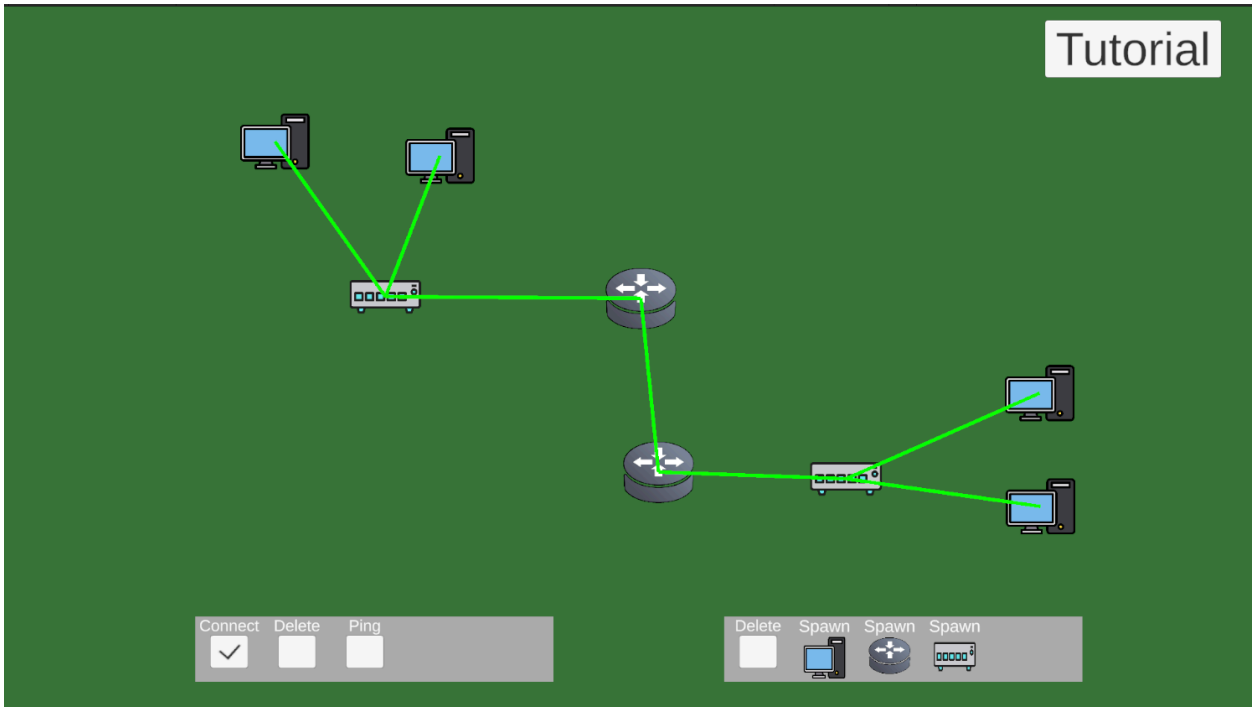

Рисунок 3.4 – Складна мережа: 2 світчі, 2 роутери та 4 ПК

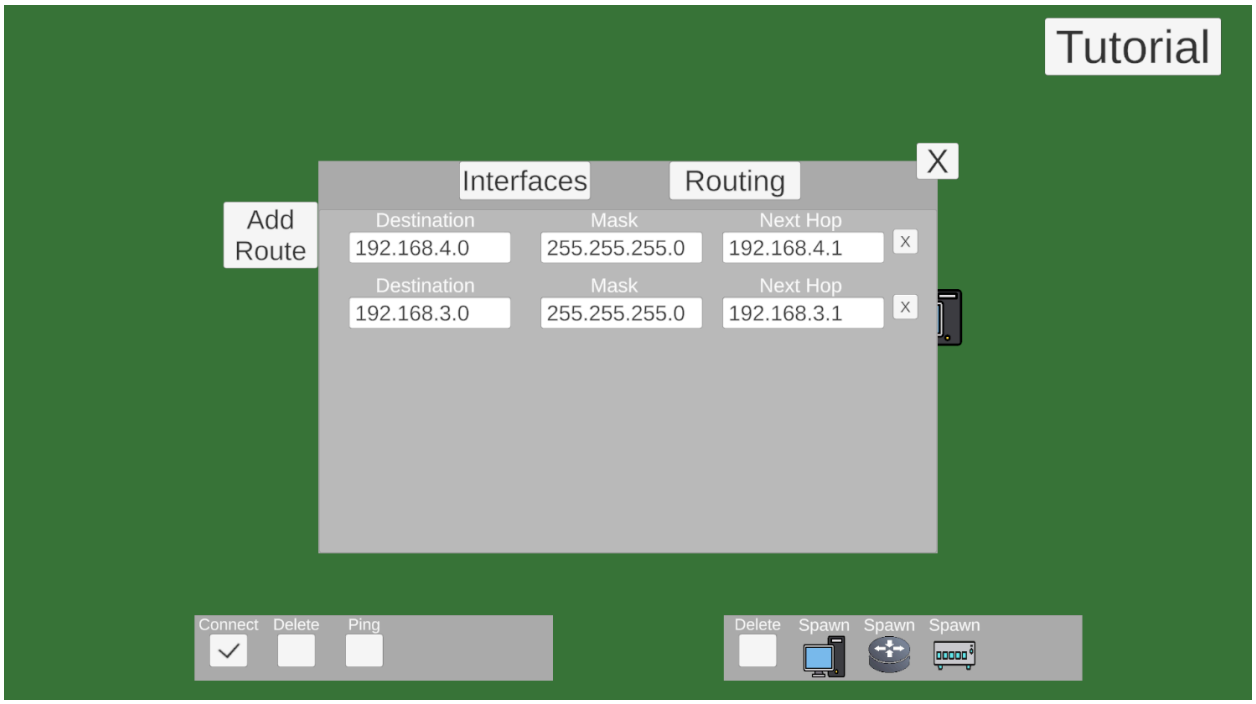

Рисунок 3.5 – Таблиця маршрутизації в інтерфейсі користувача

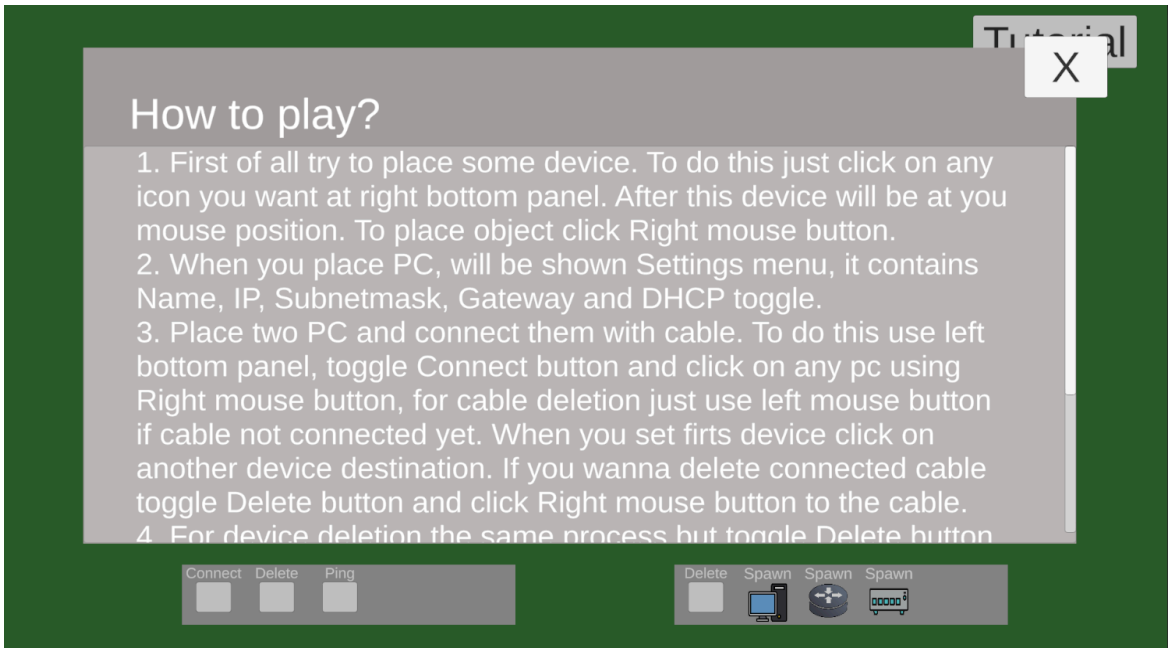

Рисунок 3.6 – Туторіал-вікно з короткою інструкцією

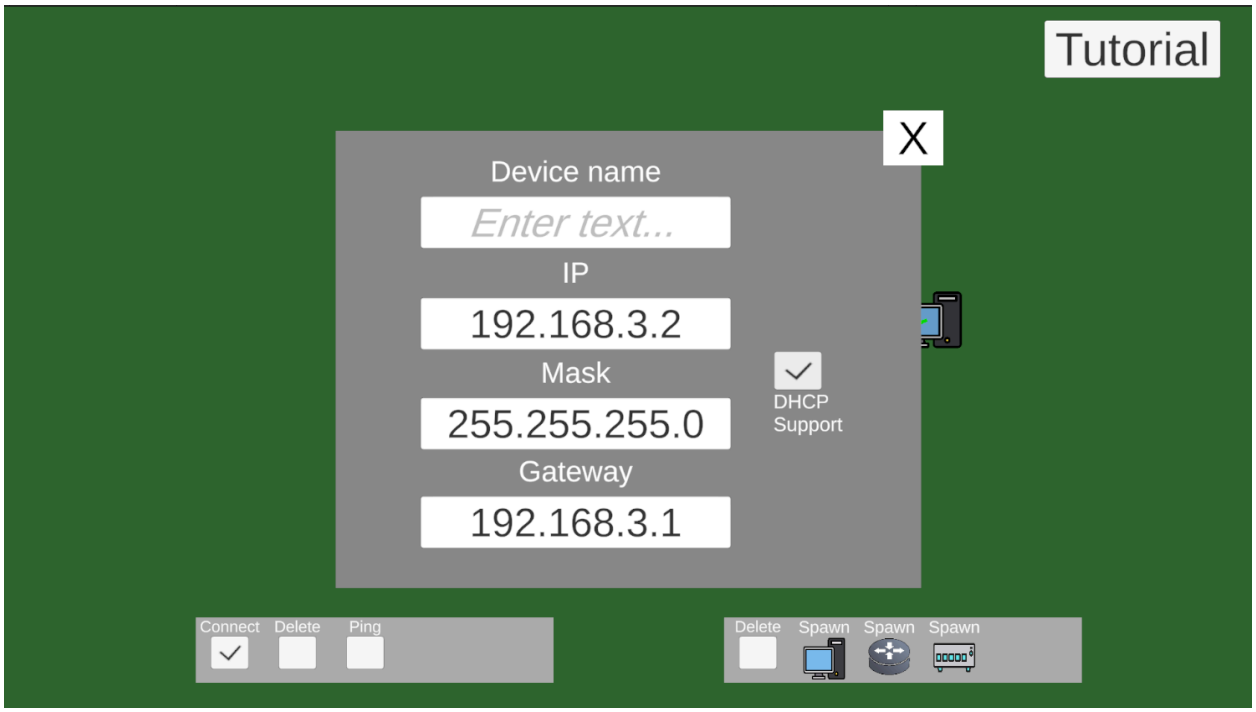

Рисунок 3.7 – Налаштування DHCP-клієнта на ПК

3.3.3.2 Редагування сцен у Unity

Редагування навчального середовища здійснювалося безпосередньо в редакторі Unity з використанням базових інструментів компонування та розміщення об'єктів на сцені. Основна ігрова сцена розроблялася з урахуванням потреб користувача в інтуїтивному середовищі, яке б дозволяло швидко орієнтуватися у просторовій структурі мережі.

Всі пристрої, що використовуються в навчальному процесі – комп’ютери, світчі, маршрутизатори – реалізовані як префаби з відповідною логікою, що інкапсулює в собі як візуальні компоненти, так і взаємодію з іншими елементами. При додаванні нового пристрою на сцену, він розміщується на шарі, зарезервованому для інтерфейсної взаємодії, що дозволяє уникати конфліктів з іншими об’єктами.

Користувач має можливість самостійно створювати конфігурації мереж, просто додаючи або видаляючи пристрої, перетягуючи кабелі або змінюючи налаштування окремих елементів. Також реалізовано кілька інтерактивних UI-елементів – кнопки та перемикачі (toggle-контролери), що активують певні функції (наприклад, режим створення кабелю або перевірки з’єднання).

Особливу увагу під час редагування сцен було приділено гнучкості – усі дії користувача в середовищі можливі без попередньої підготовки або технічних знань. Unity-сцена підтримує динамічну зміну стану об'єктів без перезавантаження або переходу в інший режим, що дозволяє зберігати ігрову плавність та безперервність навчання.

У процесі розробки активно використовувались інструменти вбудованої анімації та компонентного підходу. Наприклад, для візуалізації результатів пінгування між пристроями було використано просту анімацію кульки, що

рухається між джерелом і ціллю (рисунки 3.1, 3.8), з різним кольором залежно від результату.

Таким чином, редагування сцен у Unity стало ключовим етапом у створенні інтерактивного навчального простору, який поєднує у собі простоту, функціональність і можливість розширення у майбутньому.

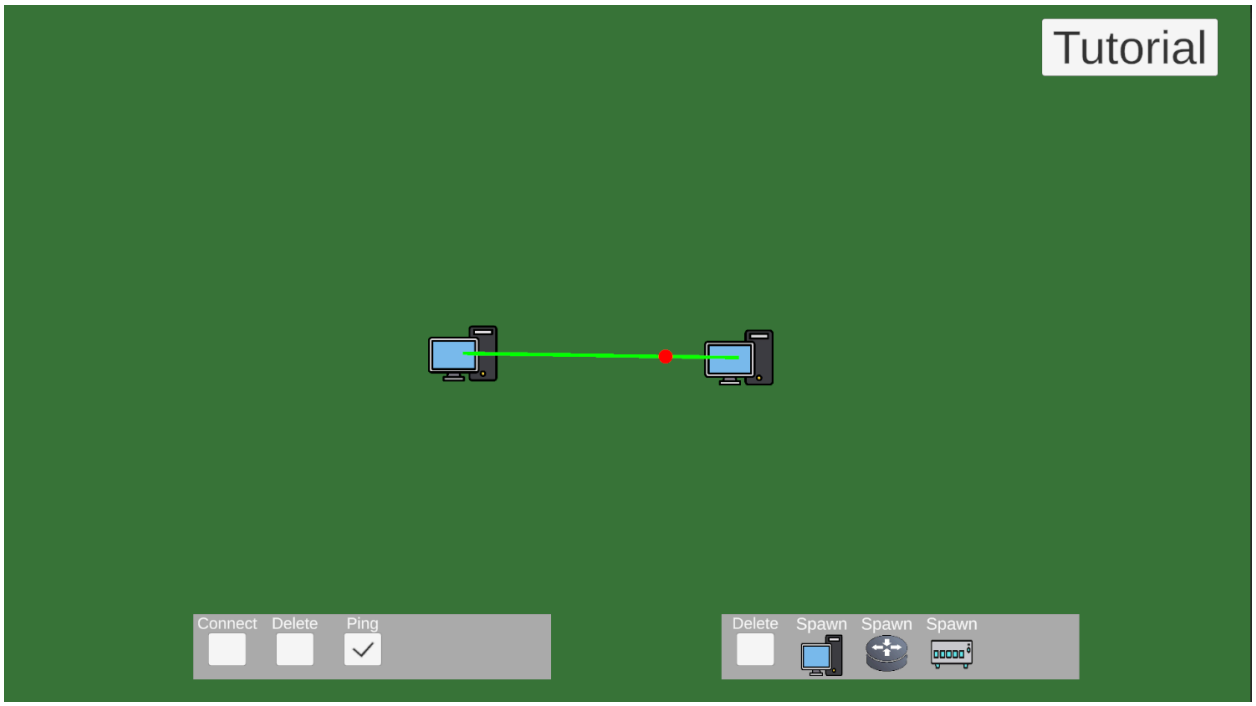

Рисунок 3.8 – Візуалізація невдалого пінгу між ПК (червоний маркер)

3.3.3.3 Переходи між сценами

У межах програмного комплексу було реалізовано два основних середовища – стартова сцена та основна навчальна сцена. Стартова сцена виконує роль головного меню, з якого користувач може розпочати навчання, ознайомитися з інструкцією або вийти із застосунку. Ця сцена містить просту структуру інтерфейсу з кнопками переходу та зображенням назви застосунку (рисунок 3.9).

Основна сцена містить усі елементи взаємодії, навчальні об'єкти та логіку поведінки користувача. При переході зі стартової сцени до навчальної завантажується середовище з інтерактивними можливостями: створенням пристроїв, прокладкою кабелів, налаштуванням параметрів мережі, а також доступом до вікна інструкцій, яке допомагає новим користувачам зорієнтуватися в управлінні (рисунок 3.6).

Переходи реалізовано засобами Unity через систему SceneManager, що дозволяє динамічно змінювати середовище на основі вибору користувача. З метою збереження плавності взаємодії, перехід між сценами супроводжується короткою анімацією або затримкою, що приховує технічні деталі процесу завантаження.

Такий підхід до структури сцени забезпечує просту логіку переходів і дозволяє при потребі масштабувати застосунок – наприклад, додати окрему сцену для перегляду досягнень або статистики користувача, не змінюючи основного архітектурного підходу.

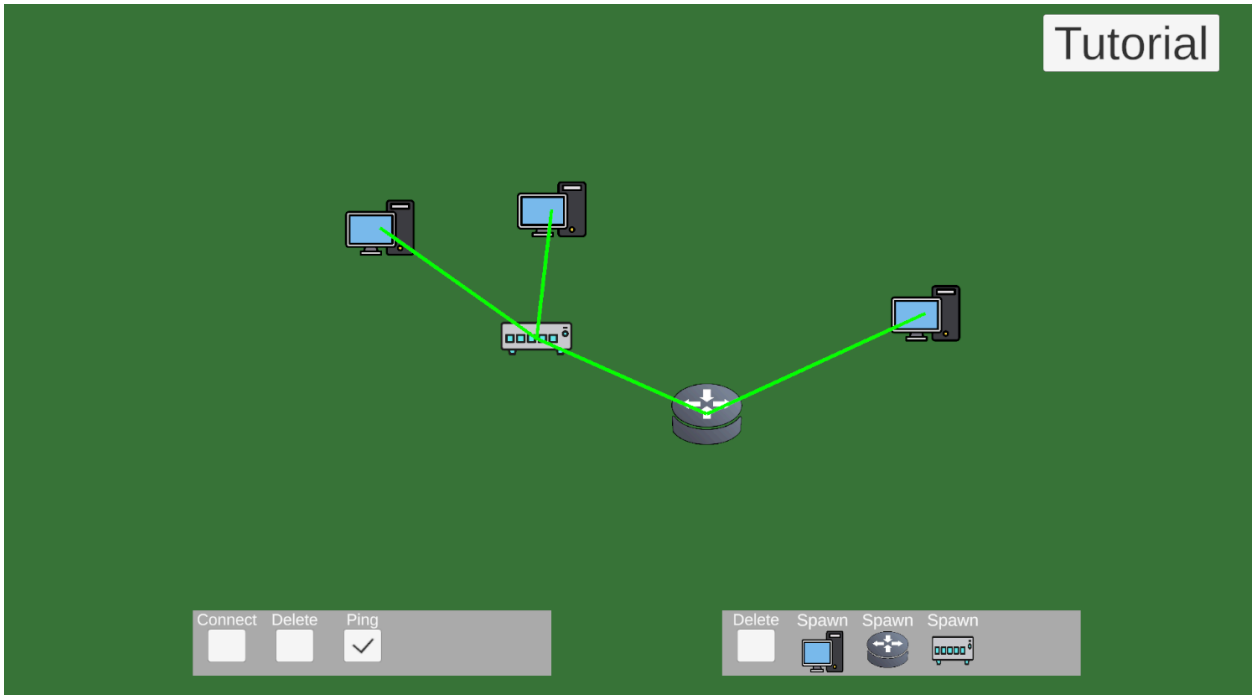

Рисунок 3.9 – Проста мережа: ПК – світч – роутер – ПК

3.3.4 Організація взаємодії користувача і системи

Користувацька взаємодія у симуляторі комп’ютерних мереж є ключовим аспектом загального навчального процесу, оскільки саме через неї реалізується пізнання, аналіз та практика. Основний фокус зроблено на інтуїтивності та легкості опанування: здобувач швидко розуміє принципи керування, отримує негайний зворотний зв’язок і має повну свободу для виконання експериментів. Такий підхід сприяє зниженню вхідного бар’єру, стимулює самостійне дослідження та активне залучення до навчального процесу.

Усі основні дії реалізуються за допомогою миші. Наприклад, для розміщення нового пристрою (ПК, світча чи маршрутизатора) достатньо натиснути відповідну кнопку інтерфейсу, після чого пристрій «прилипає» до курсора. Користувач може переміщувати його сценою і встановити натисканням лівої кнопки миші. Якщо ж передумав – натиск правою кнопкою миші скасовує дію. Подібна логіка діє і для кабелів: після вибору інструменту прокладки кабелю, користувач обирає два порти — початковий і кінцевий, — між якими створюється з’єднання (рисунки 3.1, 3.10).

Система автоматично показує лише доступні порти, спрощуючи процес підключення. Якщо порт зайнятий або несумісний, він не з’являється у вікні вибору, що виключає помилкові дії та сприяє формуванню правильних навичок. Також реалізовано простий механізм видалення пристроїв або кабелів — у процесі встановлення об’єкта натиск правою кнопкою миші відміняє дію, а для вже встановлених об’єктів доступна відповідна кнопка інтерфейсу або контекстне меню. При видаленні пристрою система також автоматично очищає пов’язані з ним порти та кабелі, що підтримує чистоту конфігурації та зменшує ймовірність помилок.

Особливу роль у взаємодії відіграють інструменти перевірки, які надають зворотний зв’язок у візуальній формі. Наприклад, під час пінгу між пристроями на сцені з’являється анімована кулька, яка рухається по кабелях, імітуючи процес передачі даних. У разі успішного проходження пакету кулька світиться зеленим, у разі помилки – червоним (рисунки 3.1, 3.8). Це не лише надає користувачеві миттєве розуміння статусу мережі, а й формує чітку асоціацію між причиною та наслідком, сприяючи глибшому засвоєнню матеріалу.

Також передбачено механізм перемикання інструментів, які визначають активну дію користувача: режим пінгу, редагування пристроїв, створення мережі тощо. Така концепція дозволяє розділяти фази взаємодії і уникати конфліктів між одночасними діями. Зокрема, якщо активний інструмент – «видалення», будь-який натиск кнопки миші буде інтерпретовано саме як намір очистити об’єкт. Це дозволяє розробити у користувача звичку до структурованої роботи, подібної до професійних систем моделювання.

Для користувачів-початківців передбачено вікно з коротким туторіалом, у якому пояснюються основні принципи взаємодії (рисунок 3.6). Це дозволяє швидко зорієнтуватися, не потребуючи зовнішніх інструкцій. Навіть без прочитання туторіалу більшість дій залишаються очевидними завдяки зручному UI, логічному розміщенню кнопок та візуальним підказкам.

У підсумку, реалізована система взаємодії поєднує простоту, гнучкість та ефективність. Вона дозволяє користувачу не лише керувати пристроями, а й розуміти логіку побудови мереж, отримувати візуальні сигнали про правильність дій і формувати стабільні практичні навички. Такий підхід значно підвищує ефективність навчання, особливо в умовах самостійного засвоєння матеріалу.

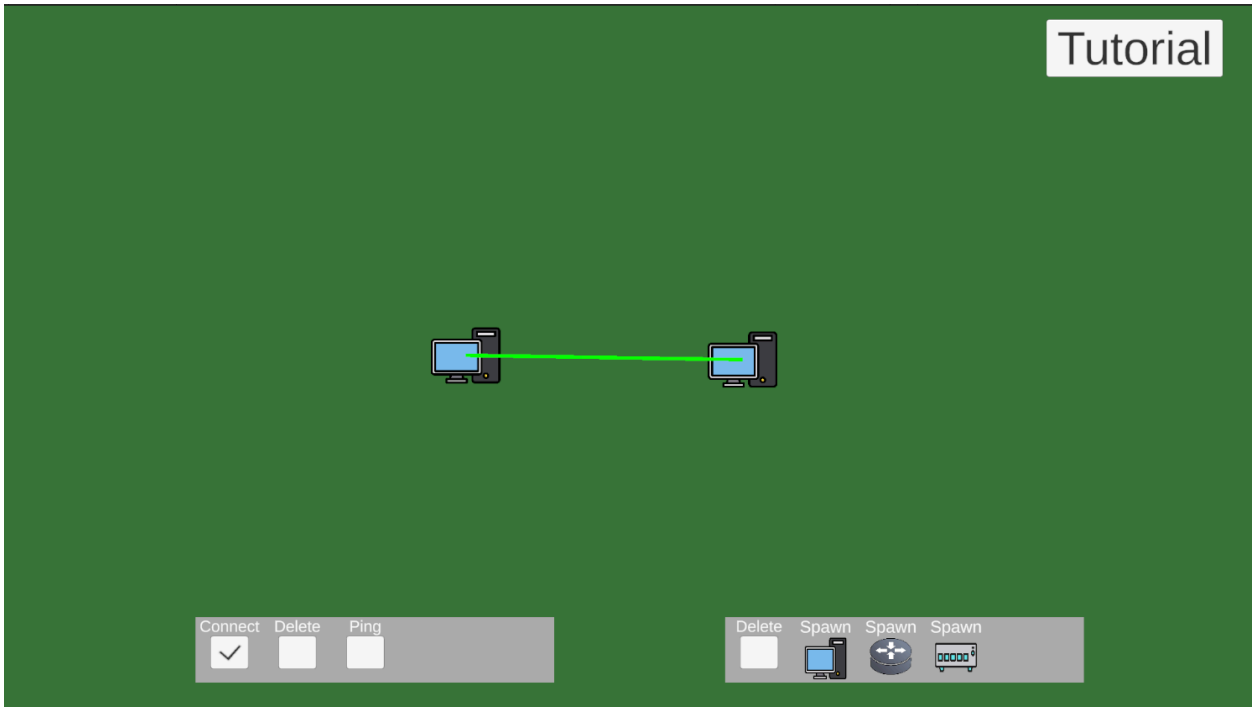

Рисунок 3.10 – Приклад мережі з двома підключеними ПК

3.3.4.1 Сценарії інтерактивності

Користувацька взаємодія в симуляторі побудована на принципах прямої маніпуляції об’єктами в середовищі. Усі дії виконуються через інтуїтивно зрозумілий інтерфейс за допомогою миші: пристрої створюються за допомогою натискання на відповідні кнопки меню, розміщуються на полі натиском правої кнопки миші, а з’єднуються кабелями шляхом протягування між портами.

У процесі створення пристрою він "слідкує" за курсором до моменту встановлення, що дозволяє зручно позиціонувати його на сцені. Якщо користувач передумав – натисканням правої кнопки миші можна скасувати створення. Така ж логіка працює і для кабелів – вони створюються правою кнопкою миші, а під час протягування можуть бути скасовані лівою кнопкою. Після встановлення пристроїв або кабелів для їх видалення передбачено окрему візуальну кнопку-перемикач, що активує режим редагування і дозволяє знищити об’єкт натиском правої кнопки миші.

Особливу роль відіграє система підключення пристроїв через порти. При натисканні для вибору з’являється pop-up список доступних портів. Зайняті порти автоматично не відображаються у виборі, що дозволяє уникнути помилок і забезпечити зручну логіку побудови з’єднань.

Користувач має змогу будувати довільні схеми – ПК ↔ ПК, ПК ↔ світч, ПК ↔ маршрутизатор, маршрутизатор ↔ світч – без обмежень, що імітує умови реального середовища з різними конфігураціями.

3.3.4.2 Механізми перевірки та підказок

Система перевірки у симуляторі реалізована через візуальні індикатори та реакцію інтерфейсу на дії користувача. Основним інструментом перевірки зв’язності виступає команда ping, яку користувач викликає шляхом вибору джерела й цілі.

Результати пінгу візуалізуються за допомогою анімованої кульки – зеленої у разі успішного з’єднання або червоної у разі невдачі. Таким чином, користувач не просто бачить текстову відповідь, а отримує наочне уявлення про маршрут проходження пакету.

У випадках, коли пінг не проходить, пакет доходить до останнього доступного вузла, а потім повертається назад червоним кольором, що дозволяє визначити точку розриву. Це імітує реальну поведінку мережевого трафіку і допомагає глибше зрозуміти принципи маршрутизації.

Підказки у вигляді текстових повідомлень або підсвічувань реалізовано лише в базових сценаріях – наприклад, при першому запуску сцени відкривається вікно-інструкція з коротким описом керування. В подальшому симулятор дає змогу самостійно досліджувати середовище, що відповідає концепції навчання через досвід і експеримент.

3.3.5 Реалізація інтерфейсу користувача

Користувацький інтерфейс є одним із ключових елементів застосунку, оскільки саме через нього здійснюється вся взаємодія користувача з навчальним середовищем. При проєктуванні інтерфейсу було дотримано принципів простоти, інтуїтивності та послідовності. Усі UI-елементи були реалізовані за допомогою стандартних компонентів Unity, що дало змогу забезпечити стабільність і контроль над адаптивністю для різних розмірів екранів.

Інтерфейс поділений на декілька логічних частин. Перша – це головне меню, яке відображається при запуску застосунку. Воно надає доступ до основних дій: запуску сесії, перегляду інструкції, налаштувань. Далі йде ігрова сцена, яка містить панелі керування кабелями, додаванням і видаленням пристроїв, а також відображенням інформації про поточні підключення. Додаткові вікна включають в себе налаштування ПК, налаштування маршрутизатора, а також окреме інформаційне вікно-туторіал, яке з’являється перед початком роботи.

Кожна панель або вікно прив’язане до відповідного діапазону дій користувача. Наприклад, при натисканні кнопки створення ПК активується режим, у якому об’єкт слідує за курсором миші до моменту встановлення. Після розміщення пристрою на сцені стає доступною взаємодія з його портами. Аналогічно реалізована логіка додавання кабелів: після вибору першого пристрою кабель починає візуально тягнутися за мишкою, і користувач має завершити з’єднання з другим пристроєм.

Усі панелі побудовані з урахуванням пропорцій 16:9. Адаптивність реалізована за допомогою системи якорів, що дозволяє інтерфейсу масштабуватися або позиціонуватись відповідно до екрану користувача.

Особлива увага була приділена зручності використання в умовах браузерного доступу: великі кнопки, чіткі підписи, візуальні підказки при наведенні та спрощена структура меню.

У результаті було реалізовано інтерфейс, що відповідає вимогам сучасного освітнього застосунку: мінімалістичний, функціональний і зрозумілий з першого погляду. Це дозволяє здобувачу зосередитись безпосередньо на навчанні, не відволікаючись на пошук необхідних кнопок чи елементів керування.

3.3.5.1 Організація UI-компонентів

Організація UI-компонентів у межах навчального застосунку базується на принципі розділення відповідальностей кожного елементу. Кожна частина інтерфейсу відповідає за окремий сегмент взаємодії – чи то запуск сесії, розміщення пристроїв, керування мережевими налаштуваннями або перевірку з'єднань. Усі елементи зібрано у відповідні панелі, які активуються або деактивуються в залежності від поточного режиму користувача.

Головне меню є першою точкою входу у застосунок. Воно містить кнопки для запуску навчальної сцени, перегляду інструкції та доступу до базових налаштувань. Цей екран представлено на рисунку 3.11. Після старту сесії користувач потрапляє до ігрової сцени, де бачить панель з інструментами для взаємодії. Тут реалізовано логіку спавну пристроїв (ПК, маршрутизаторів, світчів), вибору активних режимів (додавання кабелю, пінг, видалення) та доступу до технічних параметрів.

Для кожного пристрою передбачено окреме вікно конфігурації, яке з'являється при натисканні на відповідну кнопку. У вікні можна налаштувати IP-адресу, маску підмережі або включити DHCP. Наприклад, рисунок 3.2 демонструє вікно налаштування комп'ютера, а рисунок 3.12 – інтерфейс конфігурації маршрутизатора. Це дозволяє користувачу не лише взаємодіяти з візуальними об'єктами, а й повноцінно моделювати поведінку реальних мережевих пристроїв.

Особливу увагу приділено механізмам активації режимів взаємодії. Наприклад, користувач може натиснути на кнопку пінгу, після чого стають активними всі пристрої, доступні для вибору, і при натисканні здійснюється візуалізована перевірка зв'язку (рисунки 3.7, 3.4). Кожен режим має окремий

візуальний індикатор активності, що дозволяє уникнути непорозумінь у роботі.

Завдяки чіткій структуризації UI-компонентів застосунок залишається зручним для навігації, навіть попри розширену функціональність. Такий підхід також полегшує подальше розширення системи – наприклад, шляхом додавання нових типів пристроїв або налаштувань без необхідності переробляти весь інтерфейс.

Рисунок 3.11 – Головне меню освітнього застосунку

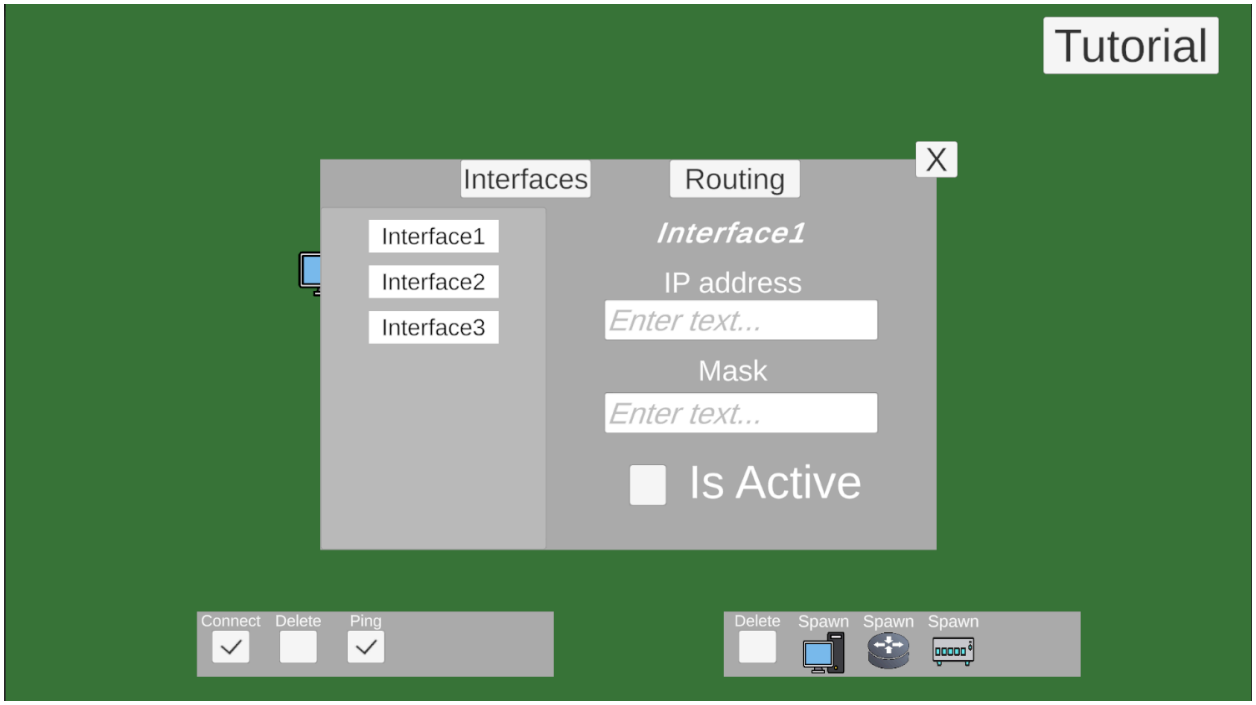

Рисунок 3.12 – Інтерфейс налаштування параметрів роутера

3.3.5.2 Адаптивність і кросплатформність

Інтерфейс навчального застосунку розроблявся з урахуванням базової адаптивності, необхідної для коректного відображення UI-елементів на екранах з різною роздільною здатністю. Основною метою було забезпечення стабільної роботи в середовищі браузера, оскільки проєкт орієнтовано на WebGL-платформу.

Усі основні UI-компоненти закріплено за допомогою системи якірців Unity, що дозволяє інтерфейсу динамічно змінювати розташування та масштаб в залежності від розміру вікна. Хоча повноцінна реалізація респонсив-дизайну не була критично необхідною для цілей кваліфікаційної роботи, передбачено підтримку широкоформатного співвідношення сторін 16:9, яке відповідає більшості сучасних пристроїв. Це дозволяє уникнути спотворень при розгортанні застосунку на екрані користувача.

Оскільки застосунок створюється виключно для роботи у браузері, специфічні адаптації під мобільні пристрої або сенсорне керування не реалізовувалися. Проте структура UI організована так, щоби в майбутньому було можливо масштабувати рішення на інші платформи, включаючи десктопні та мобільні.

Рисунок 3.6 демонструє приклад ігрового інтерфейсу в межах сцени з розгорнутими UI-панелями. Видно, що всі елементи займають логічні зони екрану – робоче поле, панель дій, меню керування. Це забезпечує зрозумілу структуру, зручність навігації та швидкий доступ до основних функцій навіть при мінімальній підготовці користувача.

Таким чином, адаптивність досягнута завдяки засобам Unity, зосередженим навколо системи якірного позиціонування, а кросплатформність реалізовано через підтримку WebGL.

3.3.6 Програмування сценаріїв взаємодії

Програмна логіка взаємодії користувача з навчальним середовищем була зосереджена на забезпеченні максимально інтуїтивного ігрового досвіду. Вся поведінка елементів реалізована засобами мови C# у середовищі Unity із застосуванням таких інструментів як Zenject для впровадження залежностей, UniTask для асинхронного програмування, R3 (ReactiveProperty) для реактивної моделі даних, а також DOTween для візуальних анімацій. Таке поєднання дозволяє створювати сучасні, модульні й легко масштабовані логіки.

Інтеракція між об’єктами відбувається у відповідності до принципу контекстної дії. Наприклад, якщо активовано режим спавну нового пристрою, то після натискання на ігрове поле пристрій закріплюється у вказаному місці. Якщо активовано режим прокладання кабелю – користувач натискає на перший пристрій, після чого кабель візуально слідує за курсором до моменту вибору другого пристрою. У момент з’єднання перевіряється коректність вибраних портів, після чого з’єднання фіксується. Така поведінка реалізована за допомогою подій, реактивного підписування та внутрішніх станів UI.

Усі дії користувача контролюються через спеціальні перемикачі (Toggle-кнопки), що активують відповідні сценарії: додавання пристроїв, видалення, з’єднання кабелем, запуск пінгу. У разі, якщо в активному стані перебуває кнопка «Ping», користувач натискає на два пристрої і запускає анімацію перевірки зв’язку між ними. Успішність або неуспішність відображається відповідною анімацією зеленої або червоної кульки, яка рухається між об'єктами, що можна побачити на рисунках 3.4 та 3.10.

Реакція на події реалізується через підписку на реактивні властивості або виклики через Unity Events, що дозволяє точно відслідковувати і

координувати всі дії гравця в рамках сесії. Також завдяки використанню Zenject підтримується поділ на логічні сервіси, які відповідають за окремі підсистеми – наприклад, керування девайсами, обробку команд пінгу, збереження прогресу тощо.

Таким чином, програмна частина взаємодії є гнучкою і масштабованою. Вона дозволяє легко додавати нові типи пристроїв або дій без втручання в існуючу архітектуру, що є критично важливим для розвитку навчального застосунку.

3.3.6.1 Логіка взаємодії з візуальними об’єктами

Уся взаємодія користувача із середовищем симуляції побудована на роботі з візуальними об’єктами, які відображають реальні мережеві пристрої та інструменти адміністрування. Основою такої взаємодії є логіка обробки подій у відповідь на дії миші, що дозволяє реалізовувати моделі натискання, перетягування, вибору і видалення об’єктів.

Наприклад, додавання нового пристрою починається з натискання на кнопку відповідного типу пристрою, після чого в сцені з'являється його візуальна копія, яка слідує за курсором. Це дозволяє користувачу попередньо побачити майбутнє розташування пристрою в мережі. Після повторного натиску лівою кнопкою миші пристрій фіксується у вибраній позиції. Якщо користувач передумав – натискання правої кнопки миші скасовує операцію. На рисунках 3.6 та 3.7 показано приклад доданих пристроїв та спроби їх з’єднання.

Для кабелів реалізовано спеціальний режим протягування: після активації користувач натискає на початковий пристрій, і система показує кабель, що тягнеться за курсором. Якщо наведено на допустимий порт іншого пристрою, з’являється спливаюче вікно з переліком вільних портів (рисунок 3.11). Успішне з’єднання візуально фіксується за допомогою LineRenderer, що створює лінію між двома пристроями.

Важливо, що у будь-який момент користувач має можливість видалити об’єкт. Це можливо як під час створення, так і після розміщення. Для цього використовується як окрема кнопка у UI, так і натискання правої кнопки миші у відповідному режимі. Наприклад, при видаленні кабелю система автоматично звільняє зайняті порти, що дозволяє уникнути помилок у подальшій конфігурації.

3.3.6.2 Обробка подій та система контролю дій

У системі взаємодії користувача з навчальним середовищем ключову роль відіграє внутрішня подієва система, яка відповідає за відстеження дій користувача та запуск відповідних сценаріїв. Такий підхід дозволяє децентралізувати логіку контролю, зберігаючи її зрозумілою та масштабованою, що особливо важливо при подальшому розширенні функціоналу.

Усередині Unity взаємодія побудована на системі подій, що виникають у відповідь на користувацькі дії, зокрема натискання кнопок, вибір об’єктів, зміна конфігурації пристроїв, спроба з’єднання чи запуск команди ping. Усі ці події генерують сигнали, які поширюються по системі і викликають відповідні реакції в інших компонентах, зокрема в UI, менеджері прогресу або модулі навчальної логіки.

Наприклад, коли користувач завершує підключення між двома пристроями, система автоматично перевіряє, чи обрані порти є вільними, та чи можлива передача даних між цими пристроями. Якщо умови виконуються, з'єднання вважається валідним, в іншому випадку користувач отримає попередження про помилку. Це дозволяє уникнути конфігураційних помилок і навчити основам коректного з’єднання в мережах.

Ще один важливий аспект – реалізація механізмів перевірки дій. У відповідь на команди пінгування система визначає маршрут до цільового пристрою, і візуалізує його за допомогою анімації руху пакету. У випадку успішного досягнення – анімація завершується зеленою кулькою, якщо ж на шляху виникла проблема – кулька повертається назад червоного кольору, наведено на рисунках 3.7, 3.4. Такий підхід дозволяє користувачу не лише побачити результат, а й зрозуміти причину помилки у конфігурації.

3.4 Очікувані техніко-економічні показники

Очікувані техніко-економічні показники розробленого програмного компонента підтверджують доцільність його створення та впровадження в освітню практику. На відміну від традиційних засобів навчання, які вимагають значних витрат на придбання фізичного мережевого обладнання, запропонований програмний симулятор дозволяє повністю реалізувати навчальні сценарії без додаткових фінансових витрат.

Економічна ефективність рішення зумовлена його веборієнтованістю, що виключає необхідність встановлення застосунку на локальні пристрої користувачів. Публікація у форматі WebGL забезпечує сумісність із широким спектром браузерів та операційних систем, що знижує витрати на обслуговування і підтримку інфраструктури. Крім того, локальне збереження конфігурацій та відсутність потреби в серверній частині мінімізують технічні витрати на експлуатацію системи.

З технічної точки зору, застосунок не потребує високопродуктивного апаратного забезпечення. Достатньо комп’ютера з базовими параметрами (4 ядра CPU, 8 ГБ оперативної пам’яті, підтримка WebGL2), що дозволяє використовувати програму у типових комп’ютерних класах навчальних закладів без додаткового оновлення обладнання.

З педагогічного погляду, очікується зростання ефективності навчального процесу за рахунок інтерактивності, візуалізації та свободи дій користувача. Гейміфікований формат та зручна система зворотного зв’язку сприяють кращому засвоєнню матеріалу, підвищенню зацікавленості та розвитку системного мислення. Таким чином, запропоноване рішення поєднує у собі технічну простоту з високою дидактичною цінністю.

4 ТЕСТУВАННЯ І ОЦІНКА ЕФЕКТИВНОСТІ

Після завершення етапу розробки важливим кроком стало тестування працездатності застосунку та оцінка його ефективності в контексті освітнього використання. Тестування охоплювало як технічні аспекти – коректність взаємодії між елементами, стабільність виконання команд, адаптивність інтерфейсу – так і користувацький досвід: зручність керування, зрозумілість навчального процесу та візуальне представлення результатів дій. Окрім цього, було проведено аналіз того, наскільки застосунок виконує свою головну функцію – навчати базовим принципам комп’ютерних мереж через інтерактивну симуляцію.

Завдяки публікації WebGL-застосунку на itch.io стало можливим провести тестування на різних пристроях та у різних браузерах. Збір зворотного зв’язку дозволив врахувати думки користувачів і внести незначні покращення в інтерфейс та сценарії взаємодії.

4.1 Методика функціонального тестування

Тестування програмного комплексу проводилося з метою перевірки коректності виконання основних функцій, стабільності взаємодії між компонентами, а також відповідності логіки роботи застосунку передбаченій освітній моделі. Оскільки застосунок є інтерактивним середовищем типу sandbox без жорстко визначеної послідовності рівнів, основна увага була приділена типово повторюваним сценаріям взаємодії.

Першочергово тестувалися загальні типові сценарії використання: створення нових пристроїв, їх встановлення у сцені, з'єднання між собою за допомогою кабелів, конфігурація налаштувань ПК та маршрутизаторів, а також запуск команди ping. Усі ці сценарії відображають ключові навчальні завдання, тому їхня стабільність мала критичне значення. Також проводилася перевірка на відповідність анімаційних і візуальних відгуків дійсним станам мережі – наприклад, успішне або неуспішне пінгування, яке має супроводжуватись анімацією зеленого чи червоного пакету.

Усі етапи тестування були виконані вручну на основі особистого використання програмного продукту. Під час проходження основних сценаріїв було виявлено ряд незначних недоліків, що стосувалися зручності використання UI-компонентів, некоректного розміщення окремих елементів при певних розширеннях екрана, а також затримок при перемиканні між режимами взаємодії. Після аналізу цих проблем були внесені відповідні зміни в інтерфейс користувача – наприклад, оновлене розміщення кнопок та підказок, поліпшена реакція на події миші при видаленні або редагуванні пристроїв (рисунок 3.4).

Оцінка ефективності тестування ґрунтується на теоретичному аналізі покриття всіх можливих варіантів взаємодії. До прикладу, перевірялась

робота мереж з різною кількістю пристроїв, складні маршрути з кількома роутерами, робота DHCP на різних комбінаціях ПК, а також сценарії, де маршрути не налаштовані – що давало змогу протестувати негативні кейси та переконатися у коректності зворотного зв’язку (рисунок 4.1).

Хоча формального юзер-тестингу не проводилося, передбачена можливість збору статистики та відгуків за допомогою аналітики платформи itch.io. Зокрема, система дозволяє фіксувати кількість запусків застосунку, тривалість сесій, популярність окремих функцій, а також показники повернення користувачів. Ці дані можуть бути використані для подальшого покращення освітньої логіки та UI-дизайну, а також адаптації складності сценаріїв.

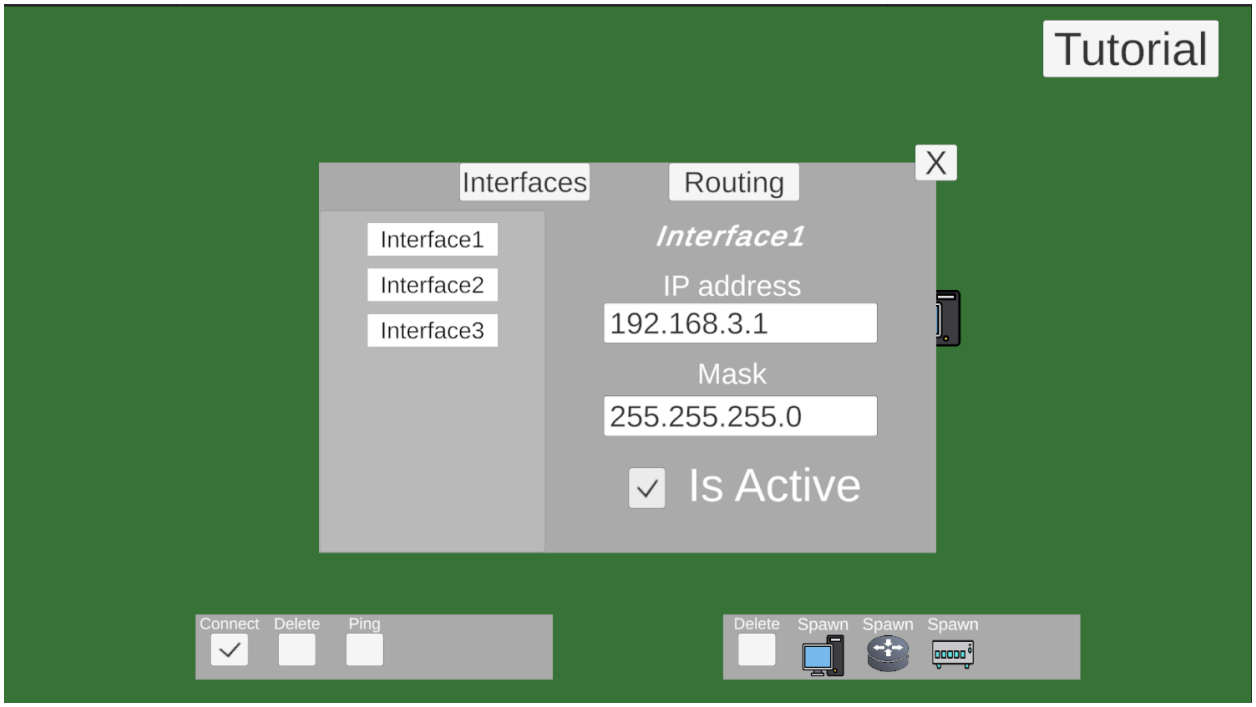

Рисунок 4.1 – Налаштований роутер після конфігурації

4.2 Аналіз зручності використання застосунку

Під час розробки програмного комплексу особлива увага приділялася забезпеченню інтуїтивно зрозумілого інтерфейсу та комфортної взаємодії з усіма елементами системи. Основним принципом була реалізація простоти в освоєнні навіть для користувачів, які не мали попереднього досвіду роботи з подібними навчальними симуляторами або інструментами мережевого адміністрування.

Початкова взаємодія відбувається через головне меню (рисунок 3.11), яке містить доступ до запуску симуляції, ознайомлення з інструкціями та базових налаштувань. Після запуску сцени відкривається вікно з туторіалом, де пояснюються основні принципи взаємодії з елементами (рисунок 3.6). Це дозволяє зменшити когнітивне навантаження на старті та сформувати правильну модель користування.

Інтерфейс управління пристроями та кабелями реалізований через логіку взаємодії миші та активних кнопок – наприклад, при натисканні кнопки спавну ПК з’являється об’єкт, що слідує за курсором, і може бути встановлений натиском миші у будь-яку позицію сцени. Це забезпечує гнучкість при побудові власної мережі. Аналогічним чином реалізована взаємодія з кабелями: після натискання на перший пристрій кабель тягнеться за мишкою до другого пристрою, що дозволяє легко візуалізувати з'єднання (рисунок 3.3.).

Важливою частиною зручності стала реалізація динамічних попереджувальних вікон при підключенні: користувач бачить лише ті порти, які доступні для підключення (рисунок 3.12). Це дозволяє уникати помилок та покращує юзер-досвід. Видалення кабелів або пристроїв під час їхнього

розміщення реалізовано простим натиском правої кнопки миші – що є природним і швидким способом корекції помилок.

Аналіз взаємодії показав, що застосунок надає логічно послідовну модель роботи, де кожна дія користувача має прямий і зрозумілий результат. Це забезпечує високий рівень зручності використання, навіть при складніших навчальних завданнях, таких як конфігурація DHCP або таблиці маршрутизації. Таким чином, загальна структура інтерфейсу та поведінка елементів у середовищі сприяють ефективному і комфортному навчанню.

4.3 Оцінка ефективності освітньої взаємодії

Ефективність освітньої взаємодії у межах розробленого програмного комплексу базується на принципі "навчання через дію". Такий підхід дозволяє здобувачам краще засвоювати матеріал, адже вони не лише читають теоретичну інформацію, а й одразу застосовують знання на практиці у симульованому середовищі.

Завдяки поетапному ускладненню завдань, реалізованому через серію інтерактивних модулів , користувачі переходять від ознайомлення з базовими поняттями до вирішення комплексних задач – наприклад, побудови мережі з кількох пристроїв і маршрутизаторів, конфігурації IP-адрес вручну та перевірки зв'язності через команду ping (рисунки 3.1, 3.8, 3.10). Така поступова структура підвищує ефективність запам’ятовування та формує глибше розуміння концепцій.

Вбудована система туторіалів та контекстних підказок (рисунок 3.6) дозволяє уникнути фрустрації на початковому етапі, що особливо важливо для користувачів без попереднього досвіду. Крім того, автоматична перевірка дій користувача забезпечує своєчасний зворотний зв’язок, який є критично важливим у процесі самонавчання.

Завдання були спроєктовані з урахуванням ключових тем, що входять до базового курсу комп’ютерних мереж: адресація, пінгування, робота протоколу DHCP, маршрутизація (рисунки 3.12, 4.1, 3.5, 3.2). Це дозволяє оцінити ефективність системи як додаткового освітнього засобу для учнів профільних технічних закладів або студентів IT-напрямів.

4.4 Збір зворотного зв’язку користувачів та аналітика itch.io

Хоча офіційного збору відгуків від сторонніх користувачів не проводилося, програмний комплекс було опубліковано на платформі itch.io, що дозволило скористатися її вбудованими інструментами аналітики для відстеження активності користувачів та базових показників взаємодії. Зокрема, було отримано можливість переглядати кількість запусків застосунку, середній час сесії, географічну розподіленість користувачів та тип пристроїв, з яких здійснювався доступ.

Аналітичні дані з itch.io дозволяють зробити попередні висновки про зацікавленість аудиторії та зручність використання застосунку. Наприклад, коротка середня тривалість сесій на стартовому етапі дозволила виявити потребу в додатковому гайд-вікні для нових користувачів (рисунок 3.6), яке й було впроваджено під час розробки.

Також спостереження за типами браузерів і розміром екранів дало змогу вдосконалити адаптивність інтерфейсу, щоби забезпечити зручне відображення UI-елементів на різних пристроях, включно з ноутбуками та десктопами з нестандартними розширеннями.

У разі продовження розробки та тестування, платформа itch.io може слугувати базовим засобом збору зворотного зв’язку: наприклад, можна активувати форму коментарів, провести опитування або інтегрувати Google Analytics для отримання глибшого розуміння поведінки користувачів у навчальному середовищі. На подальших етапах доцільно передбачити й внутрішній механізм зворотного зв’язку, що дозволить користувачам оцінювати рівні складності, залишати побажання або вказувати на помилки в логіці завдань.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено програмно-апаратний комплекс, призначений для навчання основам комп’ютерних мереж. Основною метою проєкту стало створення інтерактивного застосунку, який дозволяє користувачу вивчати мережеві принципи не лише теоретично, а й через безпосередню взаємодію з віртуальним середовищем. Завдяки поєднанню освітніх цілей із елементами гейміфікації, вдалося сформувати платформу, що стимулює інтерес, сприяє глибшому засвоєнню матеріалу та підвищує ефективність навчального процесу.

Розробка охоплювала повний цикл створення освітнього програмного продукту: від аналізу наявних рішень до проєктування архітектури, реалізації ключових функціональних модулів та проведення тестування. У застосунку реалізовано симуляцію основних мережевих процесів, зокрема вручну та автоматизоване налаштування IP-адрес, пінгування пристроїв, побудову маршрутів через таблицю маршрутизації, а також роботу протоколу DHCP. Усі ці процеси подані через інтуїтивну взаємодію з візуальними об'єктами, що ілюструють реальні компоненти мережі – комп’ютери, маршрутизатори, світчі та кабелі. Важливу роль у побудові взаємодії відіграла система подій, яка координує роботу між модулями без жорсткої залежності між ними, забезпечуючи масштабованість та гнучкість застосунку.

Функціональність застосунку охоплює сценарії створення нової мережі, динамічне з’єднання пристроїв, перевірку зв’язку між ними та реакцію системи на правильні чи помилкові дії користувача. Завдяки використанню технологій Unity, Zenject, UniTask, DOTween та R3, вдалося реалізувати

ефективну логіку взаємодії, адаптивний UI та оптимізовану продуктивність, що дозволило забезпечити стабільну роботу навіть у межах WebGL-білду. Це важливо з огляду на обраний хостинг – itch.io, який має певні технічні обмеження щодо інтеграції зі сторонніми сервісами та обробки даних.

Тестування, проведене автором, дозволило виявити й виправити недоліки, що стосувалися зручності розміщення об’єктів, візуальних підказок, а також точності та швидкодії анімацій. Було також змінено логіку керування деякими об'єктами на основі реакції користувачів під час самостійного тестування. Функціональне тестування довело, що система стабільно працює у сценаріях, наближених до реального адміністрування мережі. Розділ тестування також передбачає можливості для майбутнього розширення функціоналу на основі зібраної аналітики.

Слід зазначити, що створений застосунок не має фіксованої системи рівнів, натомість пропонує відкриту структуру середовища типу sandbox. Це дозволяє користувачу самостійно будувати власну логіку навчання, експериментувати з різними конфігураціями та багаторазово проходити один і той самий сценарій у нових варіаціях. Такий підхід стимулює аналітичне мислення, дозволяє виявляти помилки через експеримент і сприяє формуванню глибшого розуміння взаємозв’язків у комп’ютерних мережах.

У підсумку, розроблений застосунок може бути використаний як допоміжний засіб у навчальних закладах, як тренажер для самостійної підготовки або як основа для розширення системи в більш повноцінну освітню платформу з інтеграцією до LMS, онлайн-аналітики або багатокористувацької взаємодії.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ISO/IEC 7498-1:1994 – Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model.
2. Офіційна документація Unity Technologies. – https://docs.unity3d.com/
3. Zenject Dependency Injection Framework for Unity. – https://github.com/modesttree/Zenject
4. DOTween Animation Engine. – http://dotween.demigiant.com/
5. UniTask – An alternative to C# async/await in Unity. – https://github.com/Cysharp/UniTask
6. ReactiveProperty (R3) Documentation. – https://github.com/runceel/ReactiveProperty
7. itch.io – платформа для публікації ігор. – https://itch.io
8. Буров Є. В. Комп’ютерні мережі: підручник - http://lib.pnu.edu.ua/elib/local/sk/sk629495.pdf
9. Андрій Будай Дизайн-патерни - просто як двері - http://programming.in.ua/programming/basisprogramming/268-design-patterns-book-andriy-buday.html

ДОДАТОК А

Текст ігрової веб програми на базі середовища розробки Unity мовою C#

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

“ДНІПРОВСЬКА ПОЛІТЕХНІКА”

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ПАРКУВАННЯ

Текст програми

804.02070743.25025-01 12 01

Листів 23

2025

АНОТАЦІЯ

Дана програма містить у собі програмний код інтерактивного застосунку для моделювання комп’ютерних мереж у навчальному середовищі. Вона є складовою частиною програмно-апаратного комплексу, призначеного для візуалізації мережевих процесів, налаштування параметрів пристроїв, перевірки зв’язності, а також формування базових практичних навичок у сфері мережевих технологій.

Програма призначена для використання у навчальному процесі під час вивчення дисциплін з комп’ютерних мереж, а також для самостійного засвоєння основ конфігурування локальної інфраструктури. Реалізовано можливості створення мережевих топологій, налаштування IP-адресації, активації DHCP, додавання маршрутизаторів та виконання перевірки з’єднань за допомогою інструменту візуалізованого пінгу.

Програма написана мовою C# у середовищі Unity з експортом у формат WebGL та призначена для використання на комп’ютерах або ноутбуках із сучасним веббраузером, що підтримує технологію WebGL2. Установка не потрібна — застосунок запускається напряму в браузер

ЗМІСТ

Файл Computer.cs

```
using Cysharp.Threading.Tasks;
using UnityEngine;

namespace DiplomaComputerNetwork.Runtime.ComputerNetwork
{
   public class Computer : NetworkObjectBase
   {
      public override async UniTask<bool> TryReceivePacket(NetworkPacket
    packet)
      {
         await base.TryReceivePacket(packet);
         if (packet.PacketType == NetworkPacketType.Data &&
    packet.DestinationIP == IpAddress)
         {
            Debug.Log($"[Computer: {Name}] ✅ Ping SUCCESS from
    {packet.SourceIP}: {packet.Data}");
            return true;
         }
         switch (packet.PacketType)
         {
            case NetworkPacketType.DHCPOffer:
            {
               var parts = packet.Data.Split(';');
               if (parts.Length >= 5 && parts[1] == MacAddress)
               {
                  var requestedIP = parts[2];
                  var request = new NetworkPacket("0.0.0.0", "255.255.255.255",
    $"DHCPRequest;{MacAddress};{requestedIP}",
    NetworkPacketType.DHCPRequest);
                  foreach (var port in DevicePorts)
                  {
                     if (port.TryGetConnectedDevice(out var device))
                        await device.TryReceivePacket(request);
```

```
                }
            }
            break;
        }

        case NetworkPacketType.DHCPAck:
        {
            var parts = packet.Data.Split(';');
            if (parts.Length >= 5 && parts[1] == MacAddress)
            {
                SetIpAddress(parts[2]);
                SetNetworkMask(parts[3]);
                SetGateway(parts[4]);
                Debug.Log($"[{Name}] Received DHCP IP: {IpAddress}");
            }
            break;
        }
    }

    return true;
  }
 }
}
```

Файл NetworkObjectBase.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using Cysharp.Threading.Tasks;
using R3;
using UnityEngine;
using Random = UnityEngine.Random;

namespace DiplomaComputerNetwork.Runtime.ComputerNetwork
{
   public abstract class NetworkObjectBase : MonoBehaviour
   {
      [field: SerializeField] public virtual List<NetworkPort> DevicePorts { get;
    private set; }
      public virtual string Name { get; protected set; }
      public string IpAddress { get; protected set; }
      public string NetworkMask { get; protected set; }
      public string Gateway { get; protected set; }
      [field: SerializeField] public string MacAddress { get; private set; }
      public bool DHCPSupport { get; protected set; }

      public bool Interactable;

      protected virtual void Awake()
      {
         MacAddress = GenerateMacAddress();
      }
      private string GenerateMacAddress()
      {
         var bytes = new byte[6];
         for (int i = 0; i < 6; i++)
            bytes[i] = (byte)Random.Range(0x00, 0xFF);
```

```
        return string.Join(":", bytes.Select(b => b.ToString("X2")));
    }

    protected virtual void OnValidate()
    {
        Debug.Log($"🔧 [Device: {name}] OnValidate: Ports =
{DevicePorts?.Count}");
        foreach (var devicePort in DevicePorts)
        {
            devicePort.SetOwner(this);
        }
    }


    public void SetDeviceName(string deviceName) => Name = deviceName;
    public void SetIpAddress(string ip) => IpAddress = ip;
    public void SetNetworkMask(string mask) => NetworkMask = mask;
    public void SetGateway(string gateway) => Gateway = gateway;

    public void SetDHCP(bool active)
    {
        DHCPSupport = active;
        if (DHCPSupport)
        {
            TryStartDHCPHandshake().Forget();
        }
    }


    public virtual bool ValidatePacket(NetworkPacket packet) =>
        packet.DestinationIP == IpAddress && !string.IsNullOrEmpty(IpAddress)
&& !string.IsNullOrEmpty(packet.SourceIP);

    public virtual void OnPortConnected()
```

```
    {
        Debug.Log($"🔗 [Device: {GetType()}] OnPortConnected triggered");
        if (DHCPSupport)
            TryStartDHCPHandshake().Forget();
    }


    public virtual async UniTask TryStartDHCPHandshake()
    {
        var discover = new NetworkPacket("0.0.0.0", "255.255.255.255",
$"DHCPDiscover;{MacAddress}",
            NetworkPacketType.DHCPDiscover);
        foreach (var port in DevicePorts)
        {
            if (port.TryGetConnectedDevice(out var device))
                await device.TryReceivePacket(discover);
        }
    }

    public virtual async UniTask<bool> TrySendPacket(NetworkPacket packet)
    {
        if (!ValidatePacket(packet))
        {
            Debug.Log($"Packet validation failed at {Name}");
            return false;
        }

        var tasks = new List<UniTask<bool>>();
        foreach (var devicePort in DevicePorts)
        {
            if (devicePort.TryGetConnectedDevice(out var deviceBase))
            {
                tasks.Add(deviceBase.TryReceivePacket(packet));
            }
```

```
            }

            var results = await UniTask.WhenAll(tasks);
            return results.Any(result => result);
        }

        public virtual async UniTask<bool> TryReceivePacket(NetworkPacket
    packet)
        {
            Debug.Log($"[Device: {Name}] Received packet from {packet.SourceIP}
    to {packet.DestinationIP}, Type: {packet.PacketType}");
            return await UniTask.FromResult(true);
        }

        private void OnDestroy()
        {
            foreach (var port in DevicePorts)
            {
                port.TryDisconnectFromPort();
            }
        }
    }
}
```

Файл NetworkPort.cs

```
using System;
using DiplomaComputerNetwork.Runtime.ComputerNetwork.
NetworkConnection;
using R3;
using UnityEngine;

namespace DiplomaComputerNetwork.Runtime.ComputerNetwork
{
   [Serializable]
   public class NetworkPort
   {
      [field: SerializeField] public string PortName { get; private set; }
      [field: SerializeField] public NetworkObjectBase Owner { get; private set; }
      [field: SerializeField] public NetworkPort ConnectedPort { get; private set; }
      public ReactiveProperty<bool> IsConnect { get; private set; } = new();

      private CableVisual _cableVisual;

      public void SetOwner(NetworkObjectBase owner)
      {
         Owner = owner;
      }

      public void SetCable(CableVisual cableVisual)
      {
         _cableVisual = cableVisual;
      }

      public bool TryConnectToPort(NetworkPort port)
      {
         if (port == this || port.IsConnect.Value) return false;

         ConnectedPort = port;
```

```
        IsConnect.Value = true;
        port.ConnectedPort = this;
        port.IsConnect.Value = true;

        Owner?.OnPortConnected();
        port.Owner?.OnPortConnected();
        return true;
    }


    public bool TryGetConnectedDevice(out NetworkObjectBase @object)
    {
        @object = null;
        if (!IsConnect.Value) return false;
        @object = ConnectedPort.Owner;
        return true;
    }

    public bool TryDisconnectFromPort()
    {
        if (!IsConnect.Value) return false;
        if (_cableVisual != null)
        {
            UnityEngine.Object.Destroy(_cableVisual.gameObject);
        }

        ConnectedPort.IsConnect.Value = false;
        ConnectedPort.ConnectedPort = null;
        ConnectedPort = null;
        IsConnect.Value = false;

        return true;
    }
}}
```

Файл Route.cs

```
using System.Linq;

namespace DiplomaComputerNetwork.Runtime.ComputerNetwork
{
   public class Route
   {
      public string DestinationNetwork { get; private set; }
      public string SubnetMask { get; private set; }
      public string NextHop { get; private set; }

      public Route(string destinationNetwork, string subnetMask, string nextHop)
      {
         DestinationNetwork = destinationNetwork;
         SubnetMask = subnetMask;
         NextHop = nextHop;
      }

      public void SetDestinationNetwork(string destinationNetwork)
      {
         DestinationNetwork = destinationNetwork;
      }
      public void SetSubnetMask(string subnetMask)
      {
         SubnetMask = subnetMask;
      }
      public void SetNextHop(string nextHop)
      {
         NextHop = nextHop;
      }

      public bool Matches(string ipAddress)
      {
         return (IPAddressToInt(ipAddress) & IPAddressToInt(SubnetMask)) ==
```

```
                (IPAddressToInt(DestinationNetwork) &
    IPAddressToInt(SubnetMask));
      }

      private int IPAddressToInt(string ipAddress)
      {
         var segments = ipAddress.Split('.').Select(int.Parse).ToArray();
         return (segments[0] << 24) | (segments[1] << 16) | (segments[2] << 8) |
    segments[3];
      }
   }
}
```

Файл Router.cs

```
using System.Collections.Generic;
using System.Linq;
using Cysharp.Threading.Tasks;
using UnityEngine;

namespace DiplomaComputerNetwork.Runtime.ComputerNetwork
{
   public class Router : NetworkObjectBase
   {
      public override List<NetworkPort> DevicePorts =>
         RouterInterfaces.SelectMany(routerInterface =>
    routerInterface.DevicePorts).ToList();

      [field: SerializeField] public List<RouterInterface> RouterInterfaces { get;
    private set; } = new();
      public List<Route> RoutingTable { get; } = new();

      protected override void OnValidate()
      {
         foreach (var routerInterface in RouterInterfaces)
```

```
        {
            routerInterface.Router = this;
        }
    }

    public void AddRoute(Route route)
    {
        RoutingTable.Add(route);
    }

    public void RemoveRoute(Route route)
    {
        RoutingTable.Remove(route);
    }

    public override async UniTask<bool> TryReceivePacket(NetworkPacket
packet)
    {
        Debug.Log($" [Router: {Name}] Received packet from {packet.SourceIP}
to {packet.DestinationIP}");


        if (packet.DestinationIP == "255.255.255.255")
        {
            Debug.Log($" [Router: {Name}] Broadcast packet received → Skipping
routing.");
            return false;
        }

        if (packet.PacketType is NetworkPacketType.DHCPDiscover or
NetworkPacketType.DHCPRequest)
        {
            Debug.Log($" [Router: {Name}] DHCP packet received → Not handled
by router.");
```

```
            return false;
        }

        var route = RoutingTable.FirstOrDefault(r =>
r.Matches(packet.DestinationIP));
        if (route != null)
        {
            Debug.Log($" [Router: {Name}] Route match:
{route.DestinationNetwork}/{route.SubnetMask} via {route.NextHop}");

            foreach (var routerInterface in RouterInterfaces)
            {
                foreach (var port in routerInterface.DevicePorts)
                {
                    if (port.TryGetConnectedDevice(out var device) &&
device.IpAddress == route.NextHop)
                    {
                        Debug.Log($" [Router: {Name}] Forwarding packet via interface
{routerInterface.Name} to NextHop {route.NextHop} → {device.Name}");
                        return await routerInterface.TryReceivePacket(packet);
                    }
                }
            }

            Debug.LogError($" [Router: {Name}] Found route but no interface is
connected to NextHop {route.NextHop}!");
        }
        else
        {
            Debug.LogWarning($" [Router: {Name}] No route found for
{packet.DestinationIP}. Packet dropped.");
        }

        return false;  }}}
```

Файл RouterInterface.cs

```
using System.Collections.Generic;
using System.Linq;
using Cysharp.Threading.Tasks;
using UnityEngine;

namespace DiplomaComputerNetwork.Runtime.ComputerNetwork
{
   public class RouterInterface : NetworkObjectBase
   {
      public override string Name => DevicePorts[0].PortName;
      [field: SerializeField] public bool IsUp { get; private set; }

      [SerializeField] private int StartHost = 2;
      [SerializeField] private int EndHost = 254;

      private readonly Dictionary<string, string> _macToIp = new();
      private readonly HashSet<string> _leasedIPs = new();

      public Router Router;


      public override bool ValidatePacket(NetworkPacket packet)
      {
         return true; // Forward all
      }

      public void SetStatus(bool isUp) => IsUp = isUp;

      public override async UniTask<bool> TryReceivePacket(NetworkPacket
    packet)
      {
```

```
    Debug.Log($" [RouterInterface: {Name}] Received {packet.PacketType}
from {packet.SourceIP} → {packet.DestinationIP}");

    switch (packet.PacketType)
    {
        case NetworkPacketType.DHCPDiscover:
        {
            Debug.Log($" [RouterInterface: {Name}] Handling DHCPDiscover");
            var parts = packet.Data.Split(';');
            if (parts.Length < 2)
            {
                Debug.LogWarning($" [RouterInterface: {Name}] Invalid
DHCPDiscover packet format");
                return false;
            }

            var mac = parts[1];
            if (_macToIp.TryGetValue(mac, out var existingIP))
            {
                Debug.Log($" [RouterInterface: {Name}] MAC already known:
{mac} → {existingIP}. Sending offer...");
                return await SendOffer(mac, existingIP);
            }

            var newIP = GenerateNewIP();
            if (newIP == null)
            {
                Debug.LogWarning($" [RouterInterface: {Name}] No available IPs
to assign for {mac}");
                return false;
            }

            _macToIp[mac] = newIP;
            _leasedIPs.Add(newIP);
```

```
            Debug.Log($" [RouterInterface: {Name}] Assigned new IP: {mac} → {newIP}. Sending offer...");
            return await SendOffer(mac, newIP);
        }

        case NetworkPacketType.DHCPRequest:
        {
            Debug.Log($" [RouterInterface: {Name}] Handling DHCPRequest");
            var parts = packet.Data.Split(';');
            if (parts.Length < 3)
            {
                Debug.LogWarning($"[RouterInterface: {Name}] Invalid DHCPRequest packet format");
                return false;
            }

            var mac = parts[1];
            var requestedIP = parts[2];

            if (_macToIp.TryGetValue(mac, out var assignedIP) && assignedIP == requestedIP)
            {
                Debug.Log($" [RouterInterface: {Name}] Confirming IP {requestedIP} for MAC {mac}, sending DHCPAck");

                var ack = new NetworkPacket(IpAddress, "255.255.255.255",
                    $"DHCPAck;{mac};{requestedIP};{NetworkMask};{IpAddress}", NetworkPacketType.DHCPAck);

                foreach (var port in DevicePorts)
                {
                    if (port.TryGetConnectedDevice(out var device))
                    {
```

```
                    Debug.Log($" [RouterInterface: {Name}] Sending DHCPAck
to {device.Name}");
                    await device.TryReceivePacket(ack);
                }
            }

            return true;
        }

        Debug.LogWarning($"[RouterInterface: {Name}] MAC/IP mismatch
or unknown: {mac} → {requestedIP}");
        return false;
      }
    }

    return await base.TryReceivePacket(packet);
  }

  private UniTask<bool> SendOffer(string mac, string ip)
  {
    var offer = new NetworkPacket(IpAddress, "255.255.255.255",
      $"DHCPOffer;{mac};{ip};{NetworkMask};{IpAddress}",
NetworkPacketType.DHCPOffer);

    var tasks = DevicePorts
      .Where(p => p.TryGetConnectedDevice(out _))
      .Select(p => p.ConnectedPort.Owner.TryReceivePacket(offer));

    Debug.Log($" [RouterInterface: {Name}] Broadcasting DHCPOffer to
MAC: {mac}, IP: {ip}");
    return UniTask.WhenAll(tasks).ContinueWith(r =>
    {
      var any = r.Any(x => x);
      Debug.Log(any
```

```
                ? $" [RouterInterface: {Name}] DHCPOffer delivered to at least one
    device"
                : $" [RouterInterface: {Name}] DHCPOffer not delivered");
            return any;
        });
      }

      private string GenerateNewIP()
      {
        if (string.IsNullOrEmpty(IpAddress))
        {
          Debug.LogError($" [RouterInterface: {Name}] IPAddress not set! Can't
    generate DHCP pool.");
          return null;
        }

        var baseParts = IpAddress.Split('.').Select(int.Parse).ToArray();
        for (int i = StartHost; i <= EndHost; i++)
        {
          var ip = $"{baseParts[0]}.{baseParts[1]}.{baseParts[2]}.{i}";
          if (!_leasedIPs.Contains(ip))
            return ip;
        }

        Debug.LogWarning($" [RouterInterface: {Name}] No available IPs in
    DHCP pool");
        return null;
      }
    }
}
```

Файл Switch.cs

```
using System.Collections.Generic;
using System.Linq;
using Cysharp.Threading.Tasks;
using UnityEngine;

namespace DiplomaComputerNetwork.Runtime.ComputerNetwork
{
   public class Switch : NetworkObjectBase
   {
      private readonly Dictionary<string, NetworkPort> _macToPort = new();

      public override async UniTask<bool> TryReceivePacket(NetworkPacket
    packet)
      {
         Debug.Log($"[Switch: {Name}] Received packet {packet.PacketType}
    from {packet.SourceIP} → {packet.DestinationIP}");

         string mac = ExtractMac(packet.Data);

         // Learn MAC from any packet, esp. DHCPDiscover
         if (!string.IsNullOrEmpty(mac))
         {
            var sourcePort = DevicePorts.FirstOrDefault(p =>
               p.TryGetConnectedDevice(out var dev) && dev.MacAddress ==
    mac);

            if (sourcePort != null && !_macToPort.ContainsKey(mac))
            {
               _macToPort[mac] = sourcePort;
               Debug.Log($"[Switch: {Name}] MAC learned: {mac} →
    {sourcePort.Owner.Name}");
            }
         }
```

```
    if (packet.PacketType is NetworkPacketType.DHCPOffer or
NetworkPacketType.DHCPAck)
    {
        if (!string.IsNullOrEmpty(mac) && _macToPort.TryGetValue(mac, out
var targetPort))
        {
            if (targetPort.TryGetConnectedDevice(out var dev))
            {
                Debug.Log($"[Switch: {Name}] DHCP reply directed to MAC
{mac}");
                return await dev.TryReceivePacket(packet);
            }
        }

        Debug.LogWarning($"[Switch: {Name}] No MAC match for DHCP
reply, broadcasting...");
        var tasks = new List<UniTask<bool>>();
        foreach (var port in DevicePorts)
        {
            if (port.TryGetConnectedDevice(out var dev))
                tasks.Add(dev.TryReceivePacket(packet));
        }

        var results = await UniTask.WhenAll(tasks);
        return results.Any(x => x);
    }

    // For all other packets, forward to all ports except source
    var forwardTasks = new List<UniTask<bool>>();
    foreach (var port in DevicePorts)
    {
        if (port.TryGetConnectedDevice(out var connectedDevice) &&
            connectedDevice.IpAddress != packet.SourceIP)
```

```
            {
                Debug.Log($"[Switch: {Name}] Forwarding to
    {connectedDevice.Name}");
                forwardTasks.Add(connectedDevice.TryReceivePacket(packet));
            }
        }

        var forwardResults = await UniTask.WhenAll(forwardTasks);
        return forwardResults.Any(x => x);
    }

    private string ExtractMac(string data)
    {
        if (string.IsNullOrEmpty(data)) return null;
        var parts = data.Split(';');
        return parts.Length >= 2 ? parts[1] : null;
    }
  }
}
```