Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка багатокористувацької карти для гри «Підземеля і дракони» з використанням React і Nodejs

Виконав: студент IV курсу, групи СП-41
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

_____ Вітвіцький Н.В.
(підпис) (прізвище та ініціали)

Керівник _____ Стоянов Ю.М.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Стоянов Ю.М.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Петрик М.Р.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2024

РЕФЕРАТ

Розробка сайту для покращення взаємодії гравців із ведучим з використанням фреймворку React // Кваліфікаційна робота освітнього рівня «Бакалавр» // Вітвіцький Назарій Володимирович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра прогограмної інженерії, група СП-41 // Тернопіль, 2024 // С. 69, рис. – 33, табл. – 0 , кресл. – 0, додат. – 2, бібліогр. – 12.

Ключові слова: React, Nodejs, js, WebStorm, додаток, користувач, лістинг, діаграма. В першому розділі кваліфікаційної роботи проаналізовано готові програмні рішення конкурентів, розглянуто актуальність розробки додатку для кращої взаємодії користувачів .

В другому розділі кваліфікаційної роботи здійснено розробку загальної структури та вигляду додатку, спроектовано архітектуру додатку.

В третьому розділі кваліфікаційної роботи реалізовано ключові класи додатку, проведено тестування додатку.

ABSTRACT

Development of a site to improve the interaction of players with management using the React framework // Qualification work of the educational level "Bachelor" // Vitvitskiy Nazarii Volodymyrovych // Ternopil National Technical University named after Ivan Pulyu, Faculty of Computer Information Systems and Software Engineering, Department of Software Engineering . , group SP-41 // Ternopil, 2024 // C. 69, fig. – 33, tab. - 0, armchair. - 0, add. – 2, bibliography - 12.

Keywords: React, Nodejs, js, WebStorm, application, user, listing, diagram. In the first section of the qualification work, ready-made software solutions of competitors are analyzed, the relevance of the development of an add-on for better user interaction is noted.

In the second section of the qualification work, the development of the general structure and appearance of the application was carried out, the architecture of the application was designed.

In the third section of the qualification work, the key classes of the application were implemented, and the application was tested.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ,СКОРОЧЕНЬ І ТЕРМІНІВ

ПК – персональний комп’ютер.

React – відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту вебсторінки, з якими стикаються в розробці односторінкових застосунків

Node.js – платформа з відкритим кодом для виконання високородуктивних мережевих застосунків , написаних мовою JavaScript

Webstorm – інтегроване середовище розробки для JavaScript, HTML та CSS від компанії JetBrains, розроблена на основі платформи IntelliJ IDEA

React-toastify – це бібліотека для React, яка забезпечує простий спосіб додавання сповіщень у веб-застосунок. Вона дозволяє створювати сповіщення, які з'являються на екрані на короткий час, щоб інформувати користувача про певні події.

Roughjs – це бібліотека для створення малюнків і графіки з ефектом ручного креслення в HTML5 Canvas та SVG. Вона дозволяє створювати фігури, лінії, криві.

socket.io – це бібліотека для JavaScript, яка дозволяє створювати в реальному часі двостороннє спілкування між веб-клієнтами та серверами.

Хост – це користувач, який створює та керує сесією, кімнатою або зустріччю.

Канвас – це HTML-елемент, який використовується для малювання графіки за допомогою JavaScript.

ЗМІСТ

ВСТУП

У сучасному світі настільні рольові ігри, такі як «Підземелля і дракони», набувають нових форм завдяки інтеграції з цифровими технологіями. В умовах зростаючої популярності цих ігор серед різних вікових груп виникає потреба в інструментах, які дозволяють гравцям ефективніше взаємодіяти і занурюватись у ігровий процес. Одним з таких інструментів є інтерактивні багатокористувацькі карти, що можуть значно покращити досвід гравців, надаючи їм можливість візуалізувати ігровий світ та координувати свої дії.

Метою цієї роботи є розробка інтерактивної багатокористувацької карти для гри «Підземелля і дракони» з використанням сучасних технологій веб-розробки — React і Node.js. React, як популярна бібліотека для створення користувацьких інтерфейсів, дозволить створити динамічний та інтерактивний фронтенд, тоді як Node.js забезпечить швидку та ефективну роботу серверної частини. Додатково, для реалізації цього додатку будуть використовуватися socket.io, roughjs і react-toastify. Socket.io забезпечить двостороннє спілкування в режимі реального часу між клієнтами і сервером, дозволяючи гравцям відслідковувати зміни на карті, і спілкуватися в чаті. Roughjs додасть графіці і картам вигляду ручного малювання, створюючи унікальну візуальну атмосферу для ігрового процесу. React-toastify буде використовуватися для відображення сповіщень і повідомлень, інформуючи гравців про важливі події та зміни у грі.

У рамках цієї роботи планується реалізувати карту, яка дозволить гравцям в режимі реального часу бачити карту на якій проводиться ігрова сесія. Головний користувач зможе створювати об’єкти, малювати карту, обирати потрібний колір при використанні інструментів, повертати та забирати останні дії зроблені ним, очищувати карту а інші гравці відслідковувати прогрес ігрових подій та взаємодіяти один з одним через чати та інші засоби комунікації. Важливим аспектом проекту є забезпечення стабільної роботи при великій кількості

користувачів, а також можливість легко масштабувати систему при зростанні навантаження.

Розробка цього додатку включатиме створення користувацького інтерфейсу з використанням React, розробку серверної логіки на Node.js Очікується, що цей проект значно покращить ігровий досвід для гравців «Підземелля і дракони», зробивши гру більш захоплюючою та інтерактивною.

Очікується, що результат проєкту буде використовуватися на ігрових сесіях , та буде доповнюватися новим функціоналом у майбутньому, який дозволить гравцям керувати, робити певні дії своїми персонажами. Також планується, що додаток буде доступним на Android і IOS.

РОЗДІЛ 1. ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧ ДЛЯ РОЗРОБКИ ДОДАТКУ

1.1 Огляд конкурентів

Одним із конкурентів на ринку інтерактивних інструментів для настільних рольових ігор є компанія Roll20, заснована у 2012 році. Ця компанія відома своєю онлайн-платформою для проведення настільних рольових ігор, зокрема «Підземелля і дракони». Roll20 співпрацює з багатьма відомими компаніями, такими як Wizards of the Coast, Paizo та інших. Найпопулярнішим їхнім продуктом є платформа Roll20, яка має інтеграцію з різними ігровими системами та багато інструментів для ведення ігор.

Рис. 1.1 – Логотип компанії конкурента Roll 20

Roll20 є зручним та зрозумілим застосунком. Тут міститься багато інструментів, що дозволяють гравцям створювати карти, керувати персонажами, використовувати вбудовані ігрові правила та взаємодіяти через відео- і аудіо-чати. Платформа також підтримує інтерактивність і реальний час взаємодії, що є критично важливим для багатокористувацьких ігор.

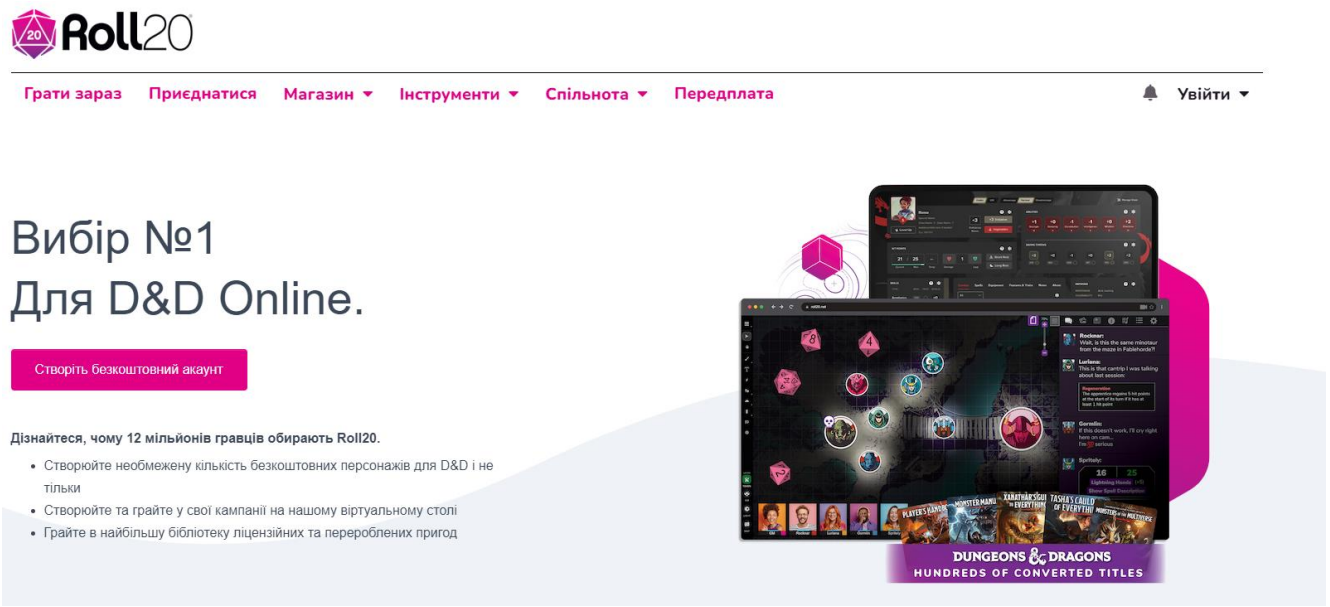

Рис. 1.2 – Вебсайт Roll 20

Дана платформа є безкоштовною з можливістю придбати преміум-функції, що дозволяє користувачам спробувати її основний функціонал перед придбанням. Однак, для деяких користувачів можуть виникнути труднощі з освоєнням всіх можливостей платформи через її багатий функціонал та складність налаштувань.

1.2 Обґрунтування вибору напрямку дослідження

Мною було обрано напрямок розробки інтерактивної багатокористувацької карти для гри «Підземелля і дракони», оскільки я зіштовхнувся з обмеженими можливостями існуючих інструментів для проведення таких ігор в онлайн-режимі. Більшість наявних рішень, хоча й функціональні, мають складні інтерфейси, які важко освоїти новачкам. Це створює труднощі для майстрів гри та гравців, особливо при організації ігрових сесій з використанням різних платформ.

Одним із таких рішень є Roll20, яке, хоча і є популярним, іноді вимагає значного часу на налаштування та освоєння. Платформа також має платні функції, що можуть бути недоступні для всіх гравців. Тому було вирішено створити додаток з більш дружнім користувацьким інтерфейсом та зручними

інструментами для проведення ігрових сесій, що дозволить зекономити час і полегшить процес організації гри.

Мій додаток буде першочергово орієнтований на зручність використання та інтерактивність, використовуючи технології React і Node.js для забезпечення швидкої та ефективної роботи. Планується розробити додаток, який включатиме основні функції для проведення ігрових сесій, з можливістю подальшого розширення функціональності. Це дозволить створити основу, яка з часом може бути доповнена новими інструментами та функціями, з урахуванням зворотного зв’язку від користувачів.

1.3 Технічний аспект проблеми

З урахуванням цих даних мною було вирішено обрати технології React та Node.js для розробки інтерактивної багатокористувацької карти для гри «Підземелля і дракони» і використання бібліотек таких, як socket.io, roughjs, і react-toastify. Цей вибір забезпечить високу ефективність додатку та зменшить споживання ресурсів, що є важливим для стабільної роботи в умовах великої кількості користувачів.

Технічні аспекти рішення. React є популярною бібліотекою для створення динамічних користувацьких інтерфейсів. Вона дозволяє розробникам створювати швидкі та інтерактивні веб-додатки з використанням компонентного підходу. Основні переваги React включають: висока продуктивність: React використовує віртуальний DOM, що дозволяє оптимізувати оновлення інтерфейсу і забезпечити високу швидкість роботи. Компонентна архітектура: Дозволяє легко розробляти, тестувати та підтримувати окремі частини інтерфейсу. Широка спільнота: Існує велика кількість бібліотек, компонентів і готових рішень, створених спільнотою, що дозволяє швидко додавати нові функціональні можливості [1].

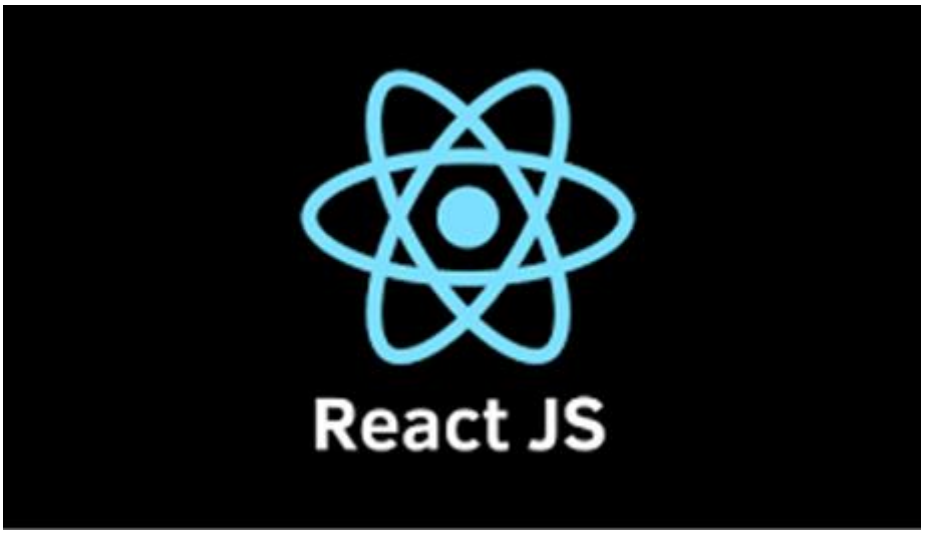

Рис. 1.3 – Логотип React js

Вибір Node.js був обраний для серверної частини додатку завдяки своїм наступним перевагам: асинхронна обробка: Node.js забезпечує асинхронну обробку запитів, що дозволяє ефективно обробляти велику кількість одночасних підключень. Швидкодія: Завдяки використанню V8 JavaScript engine від Google, Node.js забезпечує високу продуктивність. Єдиний стек технологій: Використання JavaScript для обох частин додатку — клієнтської і серверної — спрощує процес розробки та підтримки [2].

Рис. 1.4 – Логотип Node js

Socket.io є бібліотекою Nodejs тому був обраний для реалізації комунікації в режимі реального часу в додатку завдяки наступним перевагам: Двостороннє

спілкування в реальному часі: Socket.io забезпечує миттєвий обмін даними між клієнтом і сервером, що дозволяє гравцям негайно бачити редагування карти, які робить хост, і повідомлення в чаті. Socket.io легко інтегрується з такими фреймворками, як React на клієнтській стороні і Node.js на серверній стороні, що спрощує процес розробки. Проста API: Простий і інтуїтивно зрозумілий API дозволяє швидко і ефективно додавати функціональність реального часу в додаток. Підтримка простору імен і кімнат: Socket.io дозволяє організувати клієнтів у простори імен і кімнати для ефективнішого управління подіями та повідомленнями, що особливо корисно для багатокористувацьких додатків, таких як ігри..Ці переваги роблять Socket.io ідеальним вибором для створення інтерактивних багатокористувацьких додатків, де важлива швидка і надійна комунікація в режимі реального часу [3].

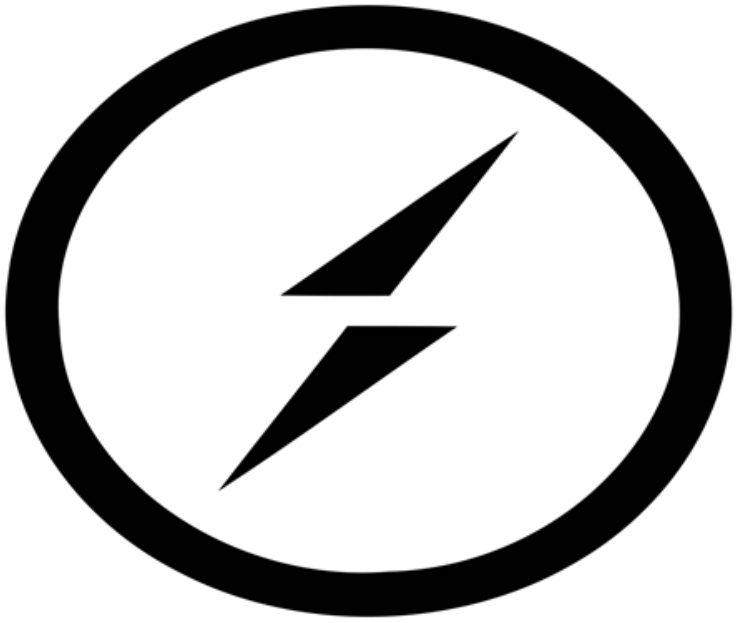

Рис. 1.5 – Логотип Socket.io

Roughjs це бібліотека яка була обрана для створення графіки з ефектом ручного креслення в додатку завдяки наступним перевагам: унікальний візуальний стиль: Roughjs дозволяє створювати графіку з виглядом ручного малювання, що додає унікальний і привабливий естетичний ефект до візуальних елементів гри. Легкість інтеграції: Roughjs легко інтегрується з різними

фреймворками, такими як React, та може бути використана з HTML5 Canvas або SVG, що робить її гнучкою у використанні. Широкий набір фігур: Бібліотека підтримує малювання різноманітних фігур, включаючи лінії, прямокутники, кола, еліпси, багатокутники та криві, що дозволяє створювати складні ілюстрації та графічні елементи.Налаштовувані параметри: Roughjs надає широкий спектр налаштувань для контролю стилю малювання, таких як ступінь грубості, товщина ліній, кольори і тіні, що дозволяє налаштовувати вигляд графіки відповідно до потреб проекту. Перфоманс: Оптимізована для високої продуктивності, Roughjs забезпечує плавне і швидке рендеринг графічних елементів навіть при високій складності сцен. Ці переваги роблять Roughjs відмінним вибором для створення візуально привабливих і унікальних графічних елементів у додатку, надаючи йому особливий стиль і покращуючи загальний ігровий досвід [4].

Бібліотека React-toastify була обрана для реалізації системи сповіщень у додатку завдяки наступним перевагам: Легка інтеграція: React-toastify легко інтегрується в будь-який React-додаток, забезпечуючи простий спосіб додавання сповіщень. Автоматичне закриття: Сповіщення можуть автоматично закриватися після заданого часу, що дозволяє уникнути накопичення великої кількості сповіщень на екрані. Різні типи сповіщень: Бібліотека підтримує інформаційні, успішні, попереджувальні та помилкові сповіщення, що дозволяє передавати користувачам різноманітні повідомлення відповідно до ситуації. Настроювані стилі: Сповіщення можуть бути налаштовані відповідно до стилю додатку, з можливістю змінювати кольори, положення на екрані, анімації та інші параметри. Підтримка анімацій: React-toastify включає вбудовані анімації для появи та зникнення сповіщень, що робить їх появу більш приємною для користувача. Сповіщення організовані в спеціальний контейнер, що полегшує їх управління та розташування на сторінці [5].

Webstorm це середовище в якому буде створюватися додаток. Воно було обране завдяки наступним перевагам: Webstorm надає зручний і потужний редактор коду з підсвічуванням синтаксису, автозавершенням коду, підтримкою численних фреймворків і бібліотек, а також інтегрованими інструментами

рефакторингу. Інтеграція з системами контролю версій: Webstorm підтримує інтеграцію з системами контролю версій, такими як Git, що дозволяє ефективно керувати змінами коду, працювати в команді та відслідковувати історію змін. Дебагінг і тестування: Вбудовані інструменти для дебагінгу і тестування дозволяють швидко виявляти і виправляти помилки, а також запускати тести для перевірки коректності роботи додатку. Інтеграція з інструментами зборки: Підтримка інтеграції з інструментами зборки, такими як Webpack, Babel, npm та інші, спрощує процес збірки та розгортання додатку. Розширюваність: Webstorm підтримує численні плагіни, які дозволяють додавати нові функції та покращувати середовище розробки відповідно до потреб проекту. Підтримка сучасних технологій: Webstorm регулярно оновлюється і підтримує останні версії фреймворків і бібліотек, що дозволяє використовувати найновіші можливості та оптимізації. Використання Webstorm забезпечить високу продуктивність розробки, зручність в управлінні проектом та полегшить процес виявлення і виправлення помилок. Це середовище розробки дозволить ефективно використовувати всі вибрані технології та бібліотеки для створення інтерактивної багатокористувацької карти для гри «Підземелля і дракони», забезпечуючи високу якість і стабільність додатку [6].

Рис. 1.5 – Логотип Socket.io

РОЗДІЛ 2. ПРОЕКТУВАННЯ ДОДАТКУ «ПІДЗЕМЕЛЛЯ І ДРАКОНИ»

2.1 Розробка загальної структури та вигляду додатку

Додаток складатиметься з двох частин.

Перша буде містити два вікна. Перше вікно називається «Create Room» воно містить в собі генерацію айді кімнати, поле для введення ім’я хоста, створення кімнати. Друге вікно називається «Join Room» воно містить в собі два поля для введення ім’я гравця, і введення айді кімнати, та кнопку для підключення до ігрової сесії.

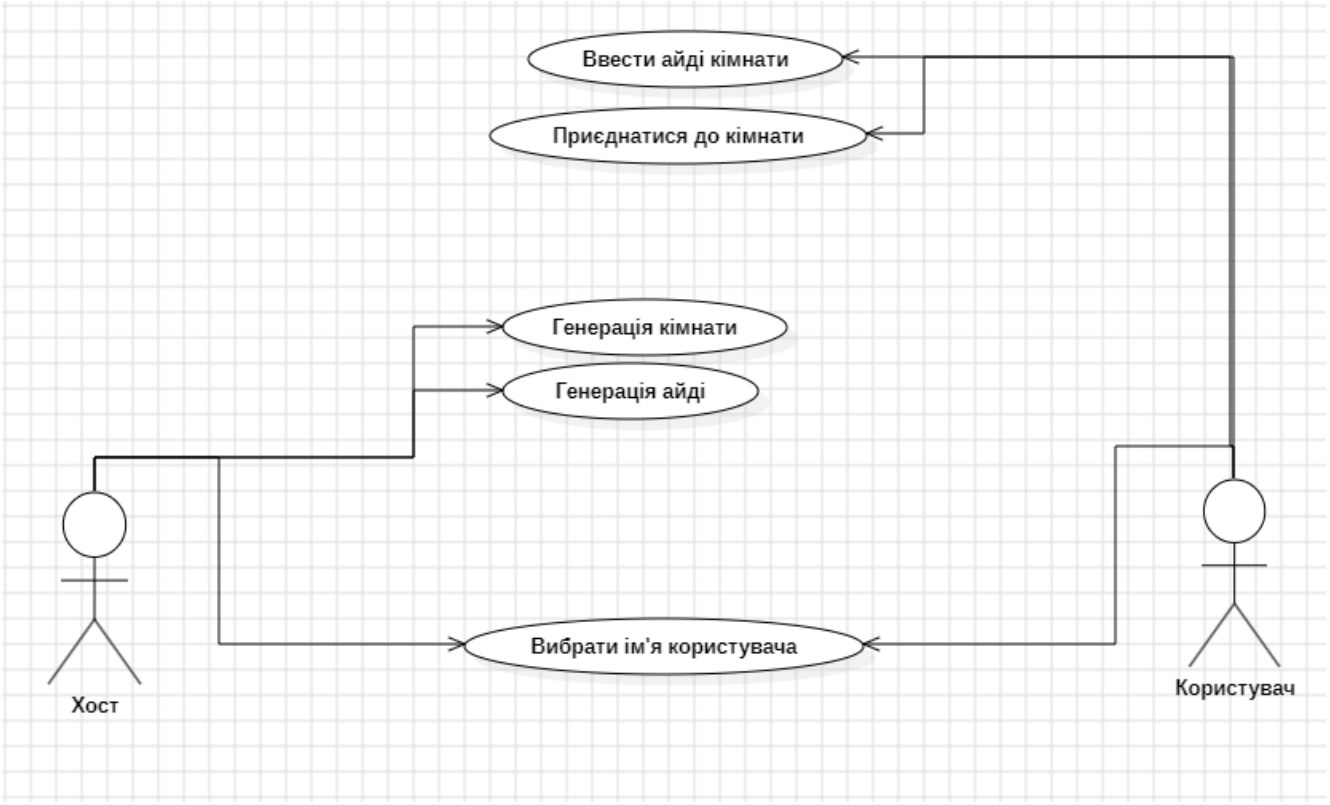

Рис. 2.1 – Діаграма варіантів використання першої сторінки вебсайту користувачем та адміністратором

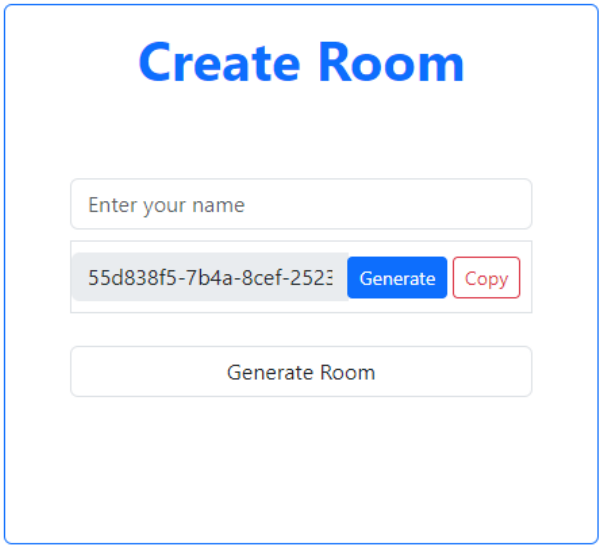

Рис. 2.2 – загальний вигляд вікна «Create Room»

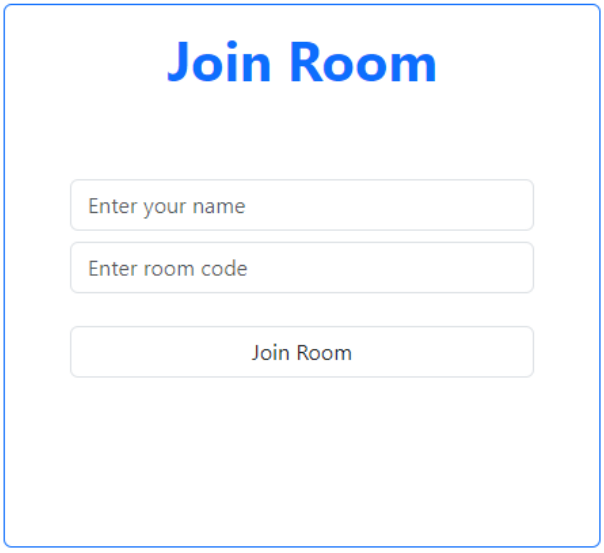

Рис. 2.3 – загальний вигляд вікна «Join Room»

Друга частина додатку поділяється на дві частини, одна від лиця користувача, а друга від хоста. Перша частина від лиця хоста містить саму кімнату з інструментами щоб конструювати карту, очищувати карту, відміняти останні зміни, і повертати останні зміни, вікно про наявних користувачів, вибір кольору для інструментів, скільки користувачів онлайн, канвас для побудови карти, та вікно чату для гравців і хоста.

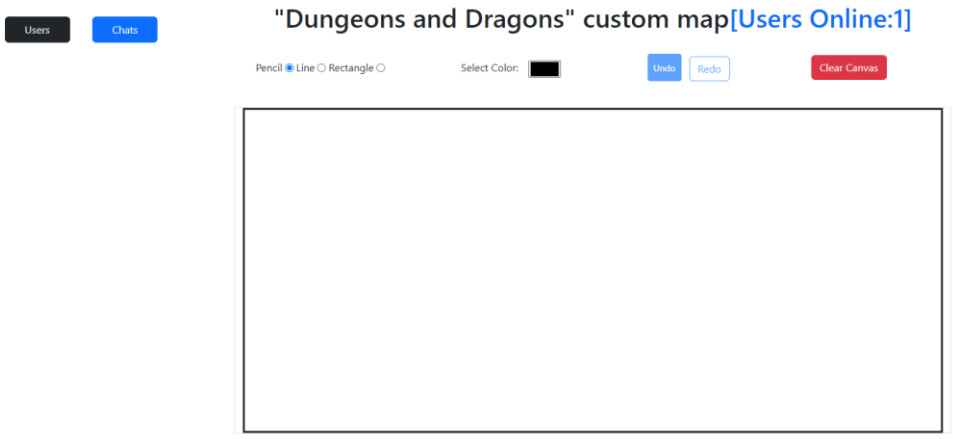

Рис. 2.4 – інтерфейс сторінки хоста

Друга частина від лиця користувача містить вікно з інформацією про наявних користувачів, канвас на якому відображається карта, скільки користувачів онлайн, та вікно чату для гравців і хоста.

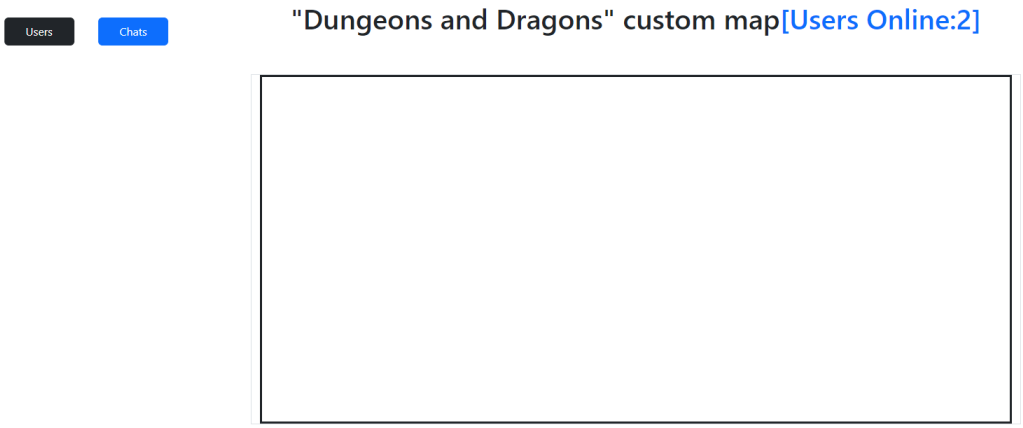

Рис. 2.5 – інтерфейс сторінки користувача

У додатку є дві ролі: користувач і хост. На рисунку 2.6 представлені можливості користувача і хоста. У користувача значно менше можливостей ніж у хоста. Хост може міняти карту під своє бачення, добавляти деталі, редагувати, відміняти останні дії, та повертати останні відмінені дії. Користувач має доступ до канвасу на якому буде карта і слідкувати за картою,бачити дії хоста. У хоста і в користувача є доступ до спільного чату в якому вони можуть комунікувати між собою або з іншими користувачами.

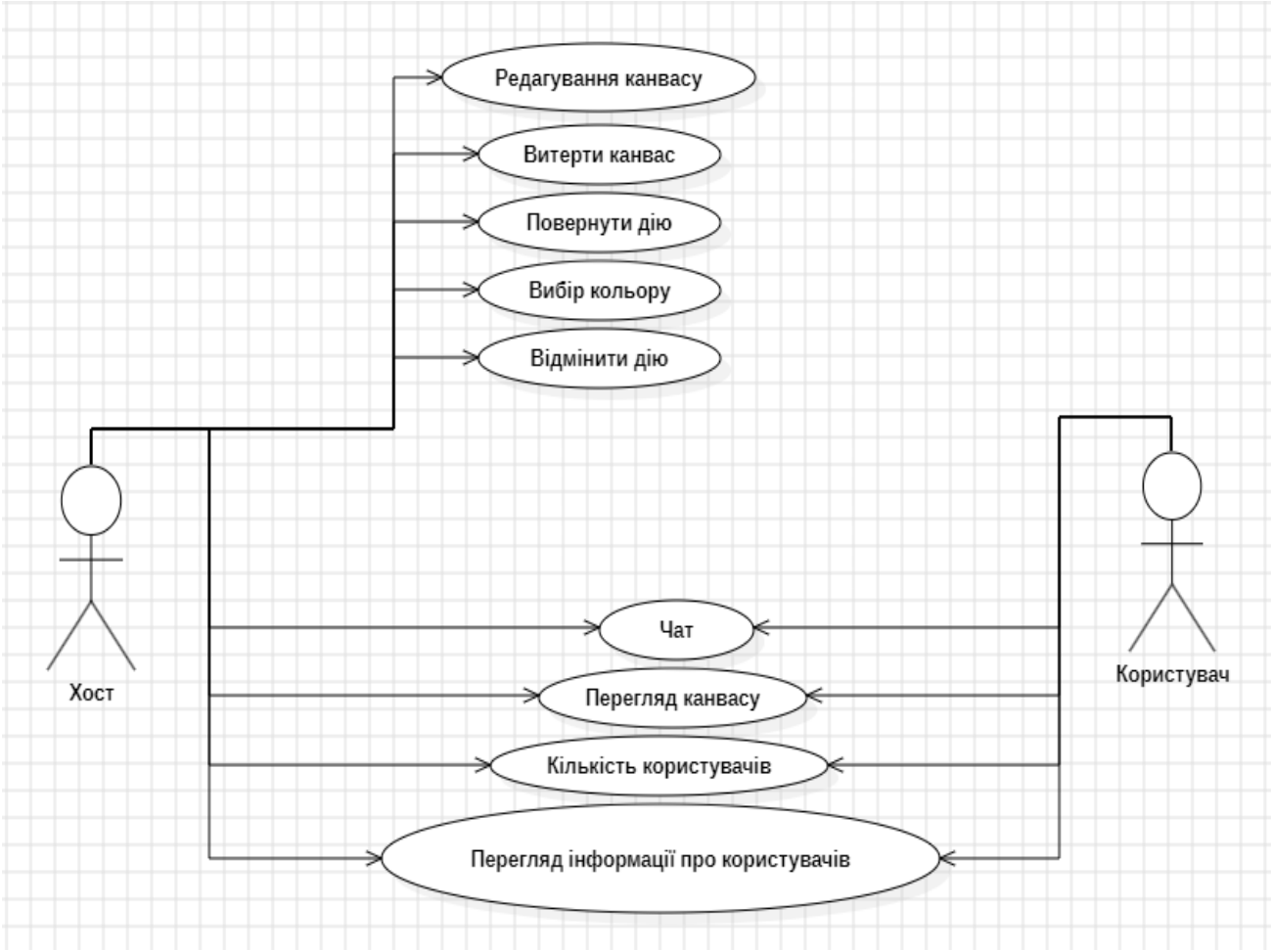

Рис. 2.6 – Діаграма варіантів використання другої сторінки вебсайту користувачем та адміністратором

2.2 Налаштування середовища розробки

Спершу потрібно налаштувати frontend для нашого проєкту. Для розробки додатку використовуватемо Webstorm [6].

Створюємо папку для нашого проєкту в даному випадку вона називається test. У webstorm переходимо в консоль і використовуємо команду npm init vite@latest [7]. Приклад встановлення на рис 2.7. Vite використовується у проєкті для налаштування сучасного середовища розробки фронтенду [8].

```
PS D:\test\frontend> npm init vite@latest
Need to install the following packages:
create-vite@5.3.0
Ok to proceed? (y) y
√ Project name: ... Diplom
√ Package name: ... diplom
√ Select a framework: » React
√ Select a variant: » TypeScript

Scaffolding project in D:\test\frontend\Diplom...

Done. Now run:

  cd Diplom
  npm install
  npm run dev

npm notice
npm notice New minor version of npm available! 10.5.2 -> 10.8.1
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.8.1
npm notice Run npm install -g npm@10.8.1 to update!
npm notice
PS D:\test\frontend>
```

Рис. 2.7 – Встановлення Vite

Тепер потрібно встановити yarn рис 2.8. Yarn використовується у проєкті як менеджер пакетів, він забезпечує швидку, надійну та безпечну роботу з залежностями, покращуючи загальний процес розробки та підтримки додатку [9].

```
PS D:\test\frontend> yarn
yarn install v1.22.22
info No lockfile found.
[1/4] Resolving packages...
[2/4] Fetching packages...
[3/4] Linking dependencies...
[4/4] Building fresh packages...

success Saved lockfile.
Done in 0.10s.
PS D:\test\frontend>
```

Рис. 2.8 – Встановлення yarn

Встановлюємо пакети roughjs і react-router-dom за допомогою команди yarn add roughjs react-router-dom необхідне для проєкту, оскільки ці бібліотеки забезпечують конкретні функціональні можливості. React-router-dom потрібен для маршрутизації, динамічного url, а roughjs для візуальних компонентів. Встановлення на рис 2.9 [4].

```
PS D:\test\frontend> yarn add roughjs react-router-dom
yarn add v1.22.22
[1/4] Resolving packages...
[2/4] Fetching packages...
[3/4] Linking dependencies...
warning "react-router-dom > react-router@6.23.1" has unmet peer dependency "react@>=16.8".
warning " > react-router-dom@6.23.1" has unmet peer dependency "react@>=16.8".
warning " > react-router-dom@6.23.1" has unmet peer dependency "react-dom@>=16.8".
[4/4] Building fresh packages...
success Saved lockfile.
success Saved 7 new dependencies.
info Direct dependencies
├─ react-router-dom@6.23.1
└─ roughjs@4.6.6
info All dependencies
├─ hachure-fill@0.5.2
├─ path-data-parser@0.1.0
├─ points-on-curve@0.2.0
├─ points-on-path@0.2.1
├─ react-router-dom@6.23.1
├─ react-router@6.23.1
└─ roughjs@4.6.6
Done in 1.85s.
PS D:\test\frontend>
```

Рис. 2.9 – Встановлення Roughjs і react-router-dom

Тепер потрібно встановити серверну частину тобто backend. З консолі потрібно зайти в папку backend і запустити команду npm init вона створить файл package.json, який знадобиться для того щоб керувати залежностями та налаштовувати проєкт.

Рис. 2.10 – Налаштування package.json

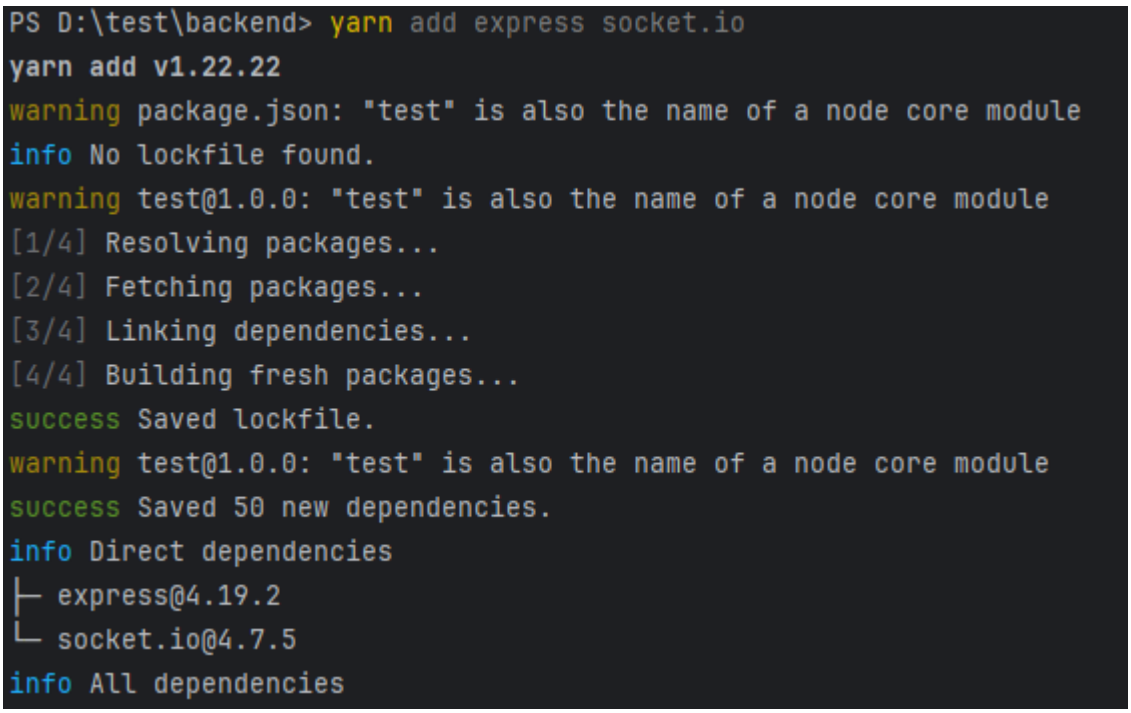

Рис. 2.11 – Встановлення бібліотек express і socket.io

Використовуємо команду yarn add express socket.io, вона використовується для встановлення двох важливих бібліотек — express та socket.io. Express — це

мінімалістичний та гнучкий веб-фреймворк для Node.js, який надає набір функціональних можливостей для створення веб-додатків і API. Socket.io надасть нам двостороннє спілкування між клієнтами [3].

2.3 Проєктування серверної частини додатку

Архітектура серверної частини проєкту побудована на основі Express і Socket.IO, що дозволяє створювати реальний час, двостороннє спілкування між клієнтом і сервером. Серверна архітектура складається з простого веб-сервера на базі Express і системи реального часу на базі Socket.IO, що дозволяє підтримувати функціонал чату та спільної роботи на білому дошці. Ключові функції, такі як управління користувачами, обробка подій приєднання/відключення, обмін повідомленнями та даними білої дошки, реалізовані за допомогою WebSocket-з'єднань [3].

2.4 Проектування архітектури

Маючи відомості про систему з попередніх розділів, почнемо проєктувати архітектуру додатку, визначимо компоненти та функції, які будуть в системі.

Отже наші компоненти будуть містити в собі функції, які будуть виконувати певні завдання. Компоненти, які використуваємо у додатку:

1. WhiteBoard
2. RoomPage
3. Forms

3.1. CreateRoomForm

3.2. JoinRoomForm

4. Chat
5. App

WhiteBoard - це компонент, який реалізує інтерактивну дошку для малювання в реальному часі.

RoomPage - це компонент, що реалізує сторінку кімнати для багатокористувацької гри "Dungeons and Dragons" з інтерактивною картою і чатом. Він включає кілька функцій і компонентів для управління інструментами малювання, кольором, історією дій, а також відображення списку користувачів і чату.

Forms - це компонент, який надає інтерфейс для створення або приєднання до кімнати в багатокористувацькій грі. Він відображає дві форми: одну для створення нової кімнати і одну для приєднання до існуючої кімнати.

CreateRoomForm - відповідає за створення нової кімнати для користувача в додатку. Він забезпечує інтерфейс для введення імені користувача та генерування унікального ідентифікатора кімнати.

JoinRoomForm - відповідає за процес приєднання користувача до існуючої кімнати в багатокористувацькому додатку.

Chat - відповідає за реалізацію функціоналу чату в багатокористувацькому додатку.

Визначившись з компонентами додатку можна буде приступити до побудови діаграми компонентів та діаграми функцій. Для побудови діаграми буде використано мову візуалізації додатків UML (Unified Modeling Language). Сьогодні UML є основним стандартом для моделювання програмних систем. Ця мова пропонує широкий набір графічних інструментів, які допомагають розробникам ефективно передавати свої ідеї та концепції. Використовуючи UML, можна створювати діаграми, які наочно відображають структуру та поведінку додатку, сприяючи кращому розумінню [10].

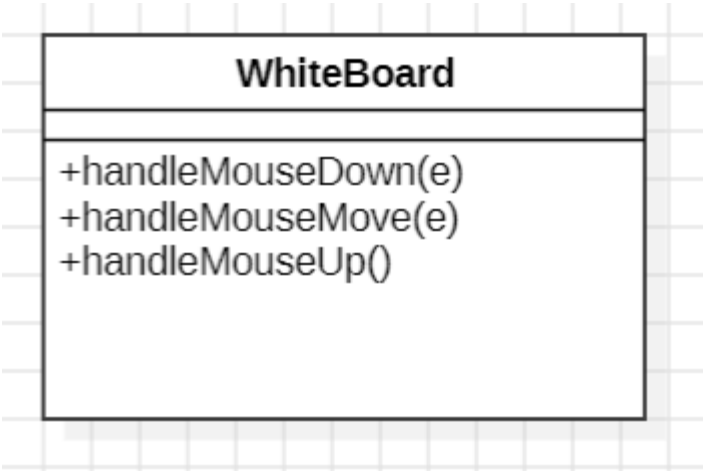

Рис. 2.12 – Компонент WhiteBoard

Компонент WhiteBoard з рисунку 2.12 є інтерактивною дошкою для малювання, яка дозволяє хосту малювати різні фігури (прямокутники, лінії та вільні лінії) та синхронізувати ці малюнки з іншими користувачами через вебсокети. Функція handleMouseDown: обробник для події натискання миші. Ініціює новий елемент малювання в залежності від вибраного інструменту. handleMouseMove: обробник для події руху миші. Оновлює координати поточного елемента малювання. handleMouseUp: обробник для події відпускання миші. Завершує поточний процес малювання. Якщо користувач не є хостом, компонент відображає зображення отримане через вебсокети. Якщо користувач є хостом, він може взаємодіяти з дошкою і малювати на ній. Принципи роботи:

1. Ініціалізація: Коли компонент монтується, він налаштовує контекст для малювання і слухає дані через вебсокети.
2. Малювання: Коли хост малює, оновлюються елементи малювання в стані, а дошка перерисовується.
3. Синхронізація: Після кожного малювання оновлене зображення дошки надсилається через вебсокети, щоб інші користувачі бачили актуальний стан дошки.
4. Відображення: Користувачі, які не є презентаторами, бачать актуальний стан дошки у вигляді зображення, яке оновлюється через вебсокети.

Цей компонент забезпечує інтерактивність і синхронізацію між користувачами, що є важливим для спільної роботи над малюнками в реальному часі.

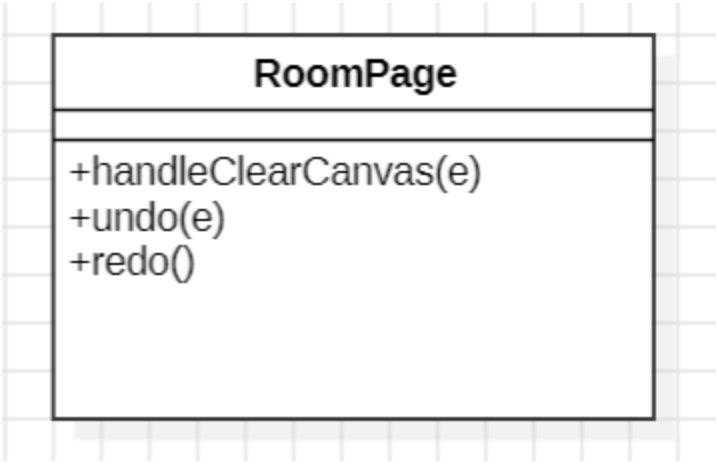

Рис. 2.13 – Копмонент RoomPage

Компонент RoomPage з рисунку 2.13 є основною сторінкою для відображення інтерактивної дошки малювання та чату в рамках сесії хоста. RoomPage має декілька функцій: handleClearCanvas очищує дошку малювання та скидає масив елементів, undo видаляє останній елемент з дошки та додає його в історію, redo відновлює останній видалений елемент з історії. Цей компонент забезпечує функціональність для управління малюванням на дошці, спілкуванням у чаті, переглядом списку користувачів, які знаходяться онлайн, а також вибір інструментів та кольорів для малювання. Компонент RoomPage інтегрує функціональність малювання, управління історією малювання, та взаємодії через чат. Він забезпечує зручний інтерфейс для користувачів, та хоста, що дозволяє їм малювати на дошці, спілкуватися та бачити інших учасників кімнати. RoomPage рендерить кнопки відкриття панелей користувачів та чату, панель користувачів (якщо відкрита) , панель чату (якщо відкрита) , заголовок з кількістю онлайн користувачів**,** **і**нтерфейс для вибору інструментів малювання, кольору, та управління історією (лише для хоста) , компонент дошки малювання. Основною

особливістю є те, що лише хост може малювати на дошці, а інші користувачі бачать оновлену версію дошки в реальному часі.

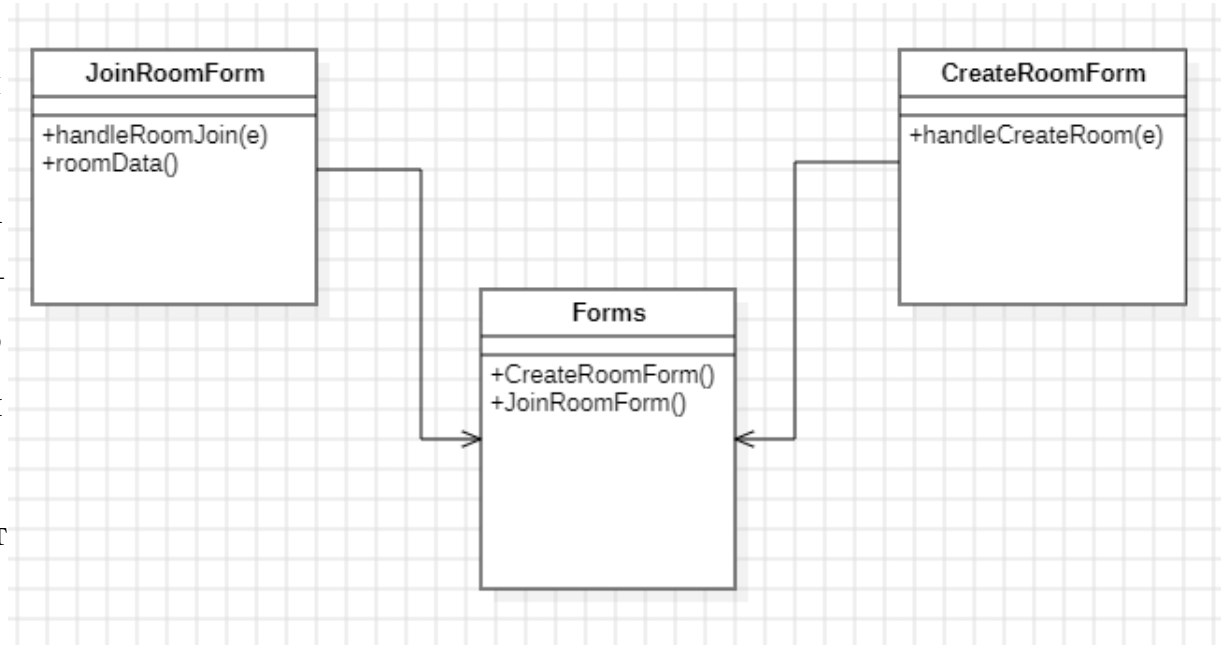

Рис. 2.14 – Компоненти Form, JoinRoomForm і CreateRoomForm

Компонент Forms з рисунку 2.14 з двома допоміжними компонентами JoinRoomForm та CreateRoomForm дозволяє користувачам створювати нові кімнати або приєднуватися до існуючих. Розглянемо кожен з цих компонентів більш детально. Компонент JoinRoomForm відповідає за створення форми для приєднання до існуючої кімнати і має дві функції handleRoomJoin обробляє подію натискання на кнопку "Join Room", створює об'єкт, і roomData, який містить інформацію про користувача і кімнату, та надсилає його на сервер. Компонент CreateRoomForm відповідає за створення форми для створення нової кімнати і має одну функцію handleCreateRoom: обробляє подію натискання на кнопку "Generate Room", створює об'єкт. Основний компонент Forms використовує JoinRoomForm та CreateRoomForm для забезпечення відповідного інтерфейсу користувача. Ці форми забезпечують простий і зрозумілий інтерфейс для створення та приєднання до кімнат, що полегшує користувачам взаємодію із системою.

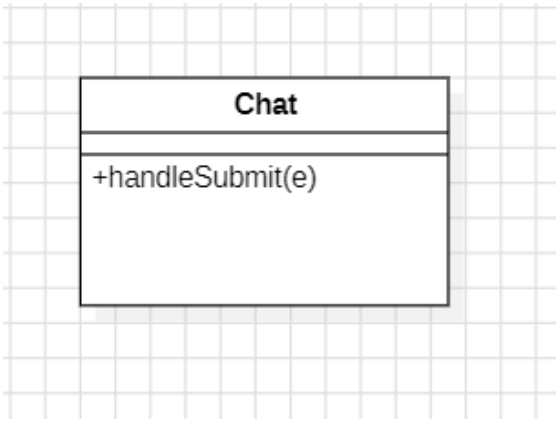

Рис. 2.15 – Компонент Chat

Компонент Chat рисунку 2.15 відповідає за функціонал чату в реальному часі, дозволяючи користувачам обмінюватися повідомленнями під час перебування в кімнаті. Chat має одну функцію обробляє подію відправки форми, додає повідомлення до локального стану та надсилає його на сервер. Цей компонент забезпечує простий і ефективний спосіб спілкування користувачів у реальному часі, інтегруючись з сервером через вебсокети для відправки та отримання повідомлень.

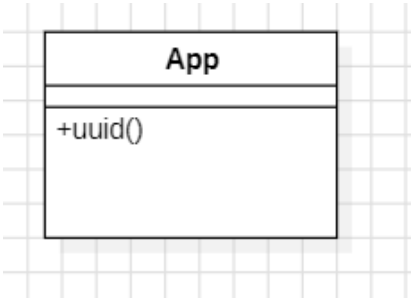

Рис. 2.16 – Компонент Chat

Компонент App є основним компонентом вашого додатку. Він відповідає за налаштування вебсокетів, управління станом користувачів та маршрутизацію між різними сторінками додатку. Імпортуються необхідні компоненти та бібліотеки, включаючи CSS-стилі, компоненти для форм та сторінок, маршрутизацію, сповіщення, хук для управління станом та бібліотеку для роботи з вебсокетами. Визначаються параметри підключення до сервера вебсокетів. Встановлює

прослуховувачі подій для вебсокетів, щоб оновлювати список користувачів та відображати сповіщення, коли користувачі приєднуються або залишають кімнату. Функція uuid генерує унікальний ідентифікатор для користувача. Компонент App служить центральним пунктом управління додатком, налаштовуючи з'єднання з сервером, обробляючи події вебсокетів, управляючи станом додатку та забезпечуючи маршрутизацію між різними сторінками.

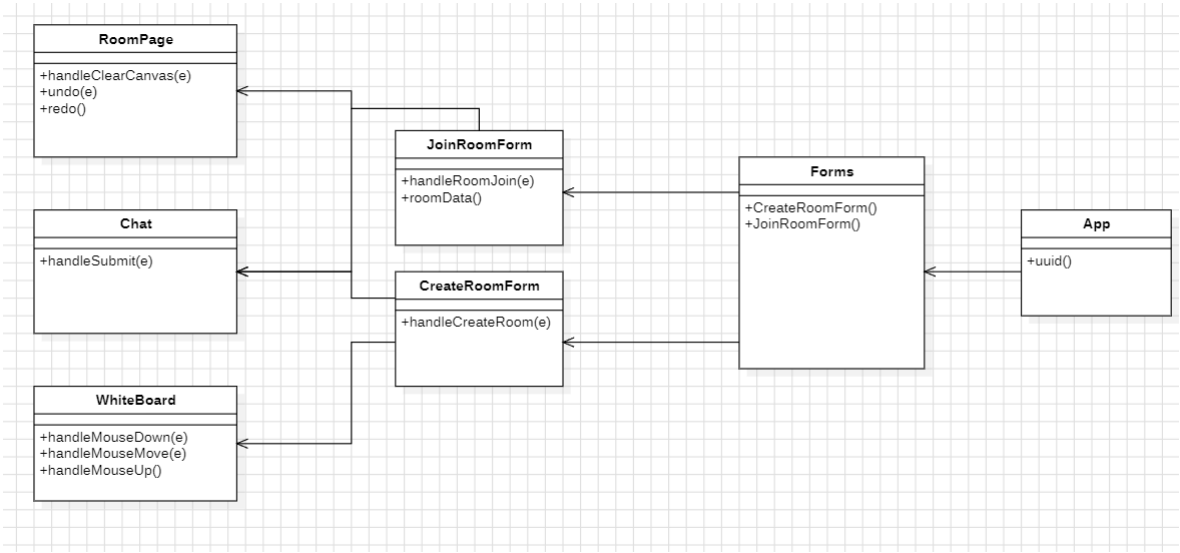

Рис. 2.17 – Діаграма компонентів додатку

На цій діаграмі відображено взаємозв'язки між основними компонентами додатку, а також компонентом Forms, який відповідає за створення та приєднання до кімнат. Зокрема, можна побачити, що більшість функціональності програми обробляється компонентами RoomPage, Chat та іншими, що успадковують функції від основного компонента App. Компонент App виступає центральним вузлом, керуючи станами користувачів, маршрутизацією та підключенням до сервера через вебсокети, забезпечуючи цілісність та інтерактивність додатку.

РОЗДІЛ 3. ТЕСТУВАННЯ ДОДАТКУ «ПІДЗЕМЕЛЛЯ ТА ДРАКОНИ»

3.1 Тестування основного функціоналу додатку

Тестування основного функціоналу додатку передбачає перевірку всіх ключових компонентів і їх взаємодії, щоб забезпечити стабільну і коректну роботу. Тестування додатку є важливим елементом у створенні продукту високої якості. Це допоможе знайти недоліки проєкту, та виправити їх. Основними аспектами тестування є:

1. Підключення користувачів:
    - Перевірка можливості користувачів приєднуватися до кімнати.
    - Перевірка правильності оновлення списку користувачів при приєднанні нових учасників.
    - Переконання, що повідомлення про приєднання нового користувача надсилаються всім учасникам кімнати.
2. Функціональність білої дошки:
    - Тестування можливості малювання на білій дошці різними інструментами (олівець, лінія, прямокутник).
    - Перевірка синхронізації малюнків між усіма учасниками кімнати в реальному часі.
    - Перевірка функцій "Undo", "Redo" та очищення дошки.
    - Перевірка вибору кольору.
3. Генерація айді
    - Перевірити чи кнопка generate code генерує айді
4. Генерація кімнати
    - Переконання що кімната створюється.

Спершу перевіримо чи коректно хост може створювати кімнату. Заходимо на вебсайт, вводимо ім’я хоста, і натискаємо на кнопку Generate Room.

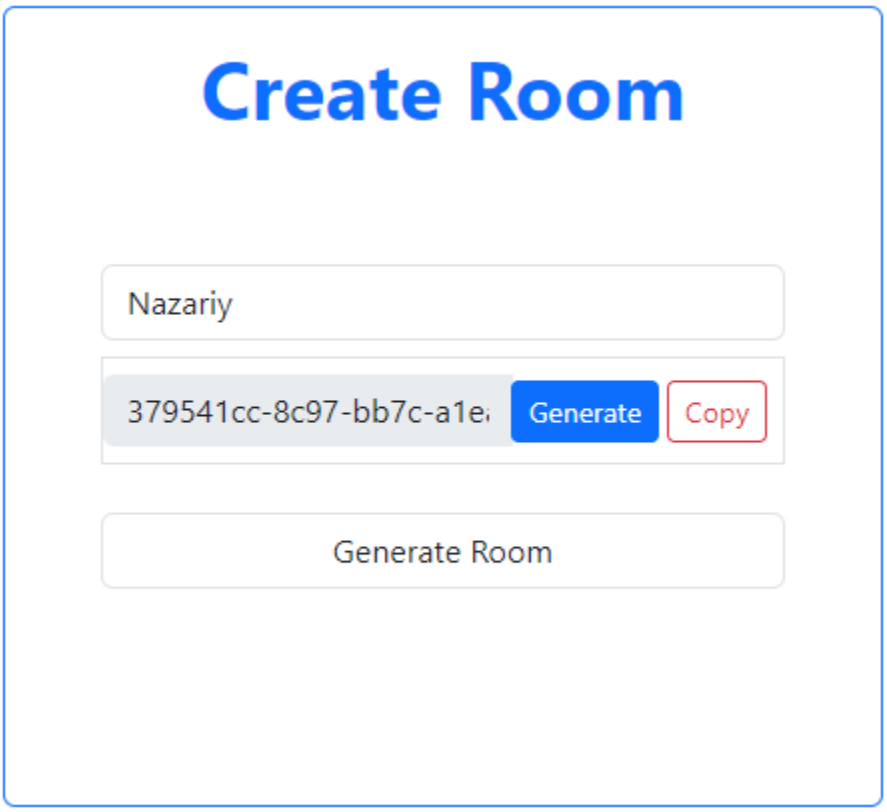

Рис. 3.1 – Тестування створення кімнати

Спершу перевіримо чи коректно хост може створювати кімнату. Заходимо на вебсайт, вводимо ім’я хоста, і натискаємо на кнопку Generate Room. Тест пройшов успішно, хоста переадресувало на створену кімнату результат на рис 3.2

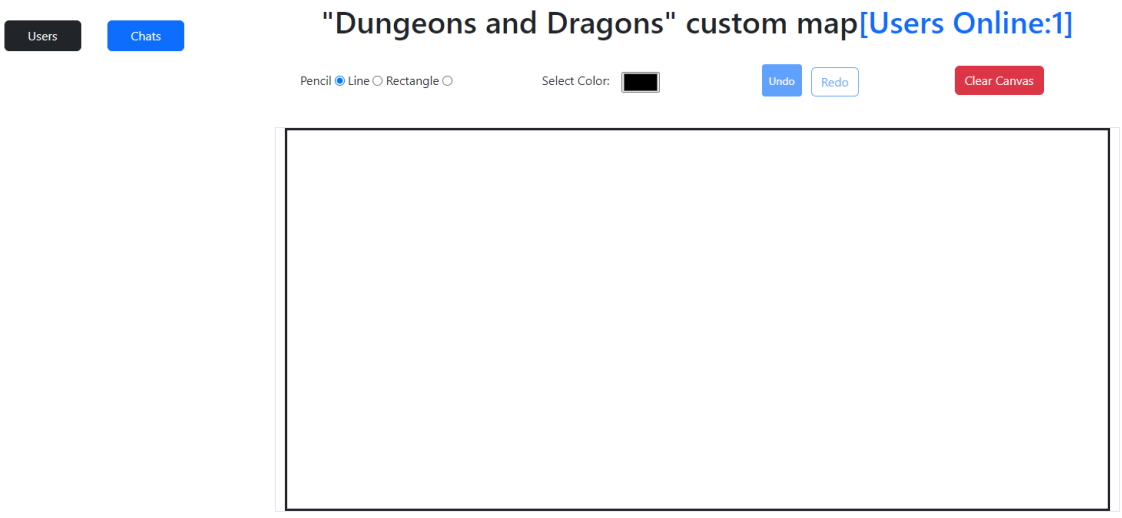

Рис. 3.2 – Вдале тестування створення кімнати

Наступним тестом буде перевірка підключення користувачів, спершу потрібно перевірити можливість користувачів приєднуватися до кімнати, ппотім потрібно перевірити правильність оновлення списку користувачів при приєднанні, і перевірити чи надсилається повідомлення про нового користувача.

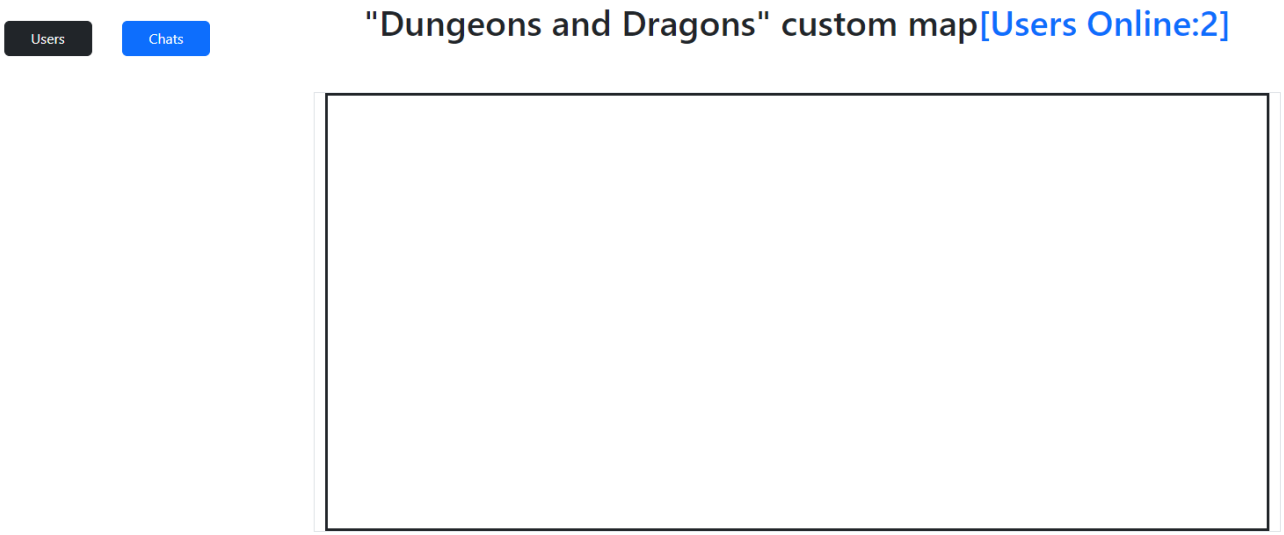

Рис. 3.3 – Приєднання до кімнати від лиця користувача

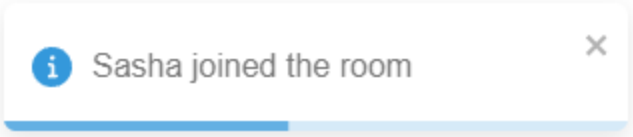

Рис. 3.4 – Повідомлення про нового користувача

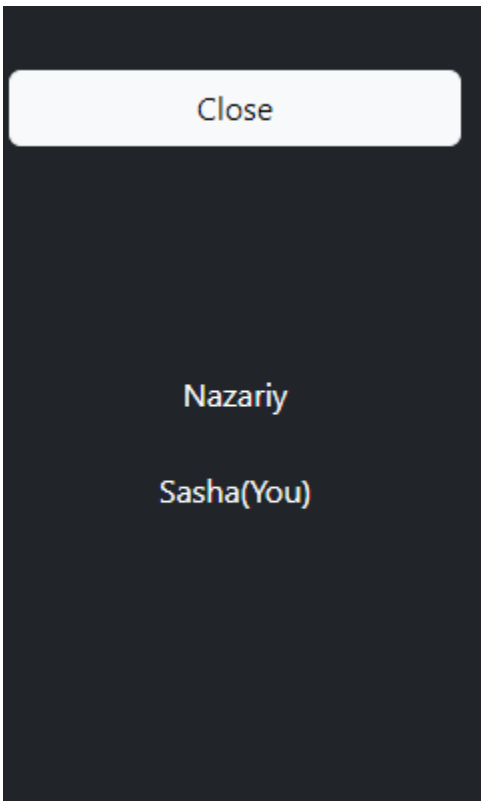

Рис. 3.5 – Вікно з інформацією про користувачів

Третій тест вимагає перевірки функціональність canvas. Потрібно перевірити чи обирається колір, який буде використовуватися для малювання, чи правильно малюють інструменти, перевірити кнопки undo, redo, clear canvas, і синхронізацію карти між учасниками групи.

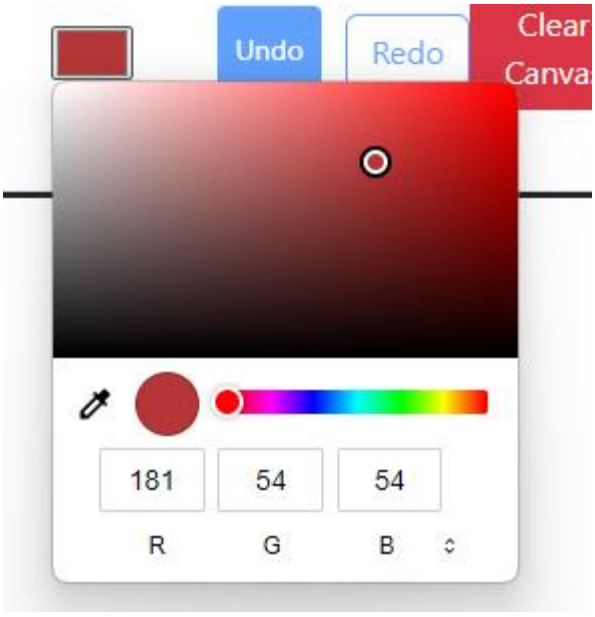

Рис. 3.6 – Вікно вибору кольору

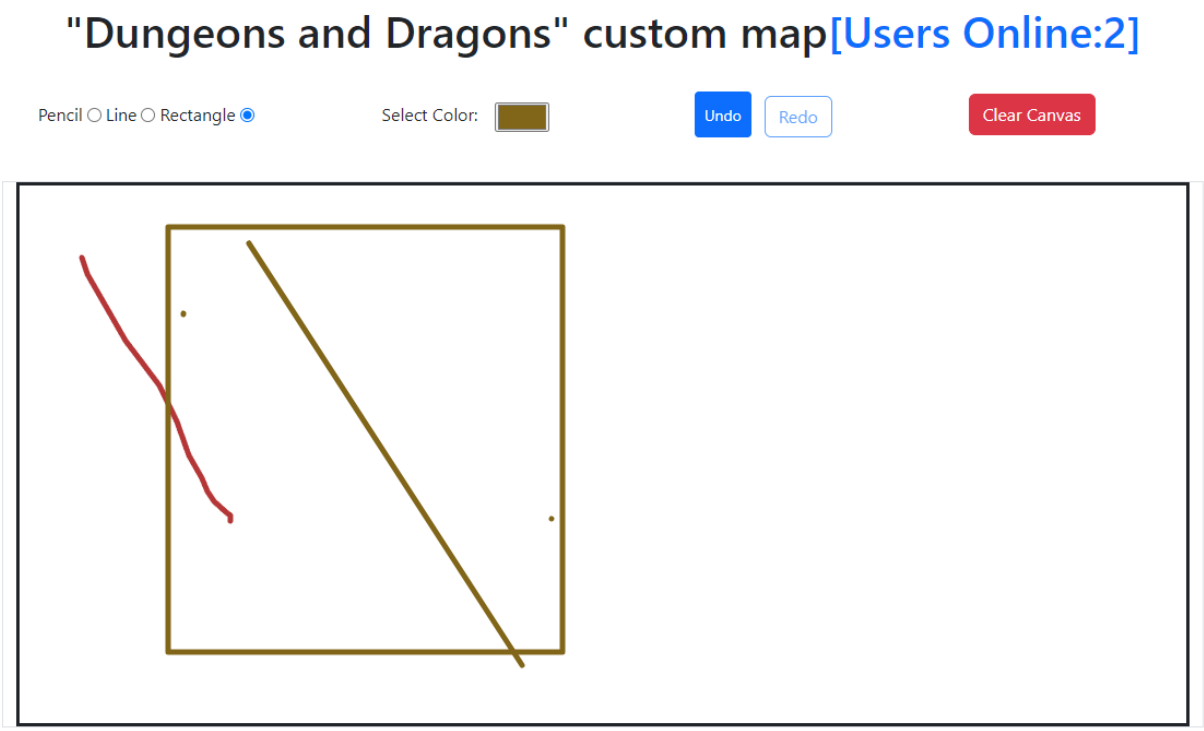

Рис. 3.7 – Результат використання інструментів

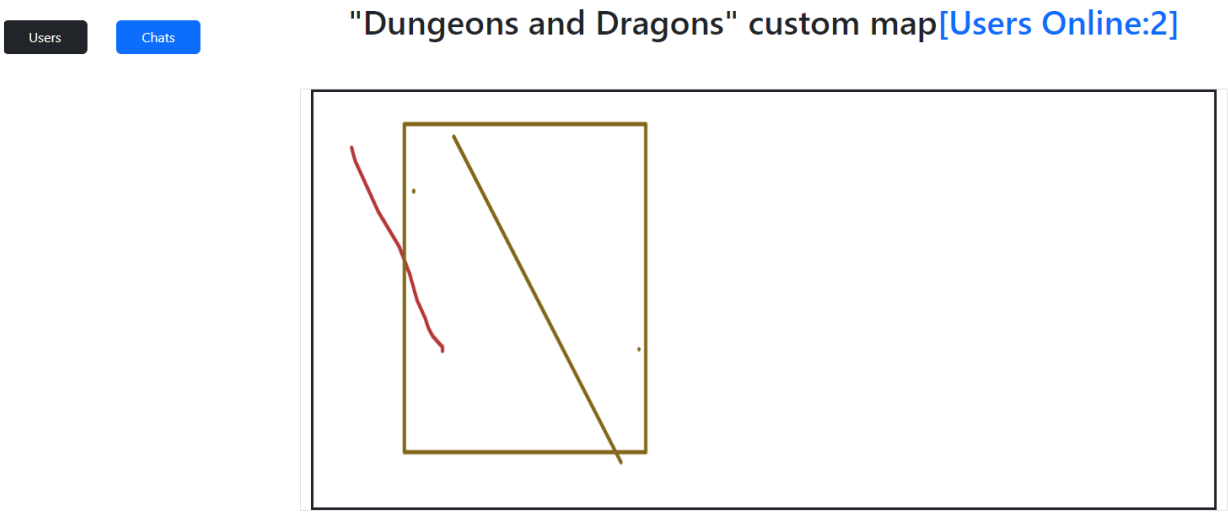

Рис. 3.8 – Синхронізація малютку від лиця користувача

Також тест із кнопками undo, redo, clear canvas пройшов успішно, кнопки виконують задане завдання.

Останній тест потребує кнопки генерації унікального айді кімнати. Хост має натиснути на кнопку generate code і йому має згенерувати айді.

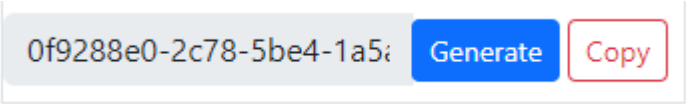

Рис. 3.9 – Автоматично згенероване айді

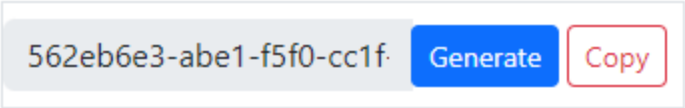

Рис. 3.10 – Згенероване айді після того як хост натиснув на кнопку генерації

Отож, враховуючи всі пройдені тест-кейси ми можемо дійти до висновку, що програмний застосунок виконується згідно заданих вимог, які були поставлені в ході роботи над вебсайтом.

3.2 Висновки третього розділу. Можливості розвитку веб застосунку

Тестування основного функціоналу додатку показало, що всі ключові компоненти працюють стабільно та коректно. Користувачі можуть приєднуватися до кімнати, малювати на білій дошці, користуватися чат-кімнатою та створювати унікальні кімнати за допомогою згенерованих айді. Тести показали, що:

1. Процес підключення користувачів:
    - Користувачі можуть без проблем приєднуватися до кімнати.
    - Список користувачів оновлюється коректно при приєднанні нових учасників.
    - Повідомлення про нового користувача надсилаються всім учасникам кімнати.
2. Функціональність білої дошки:
    - Вибір кольору для малювання працює належним чином.
    - Інструменти для малювання (олівець, лінія, прямокутник) працюють коректно.
    - Функції "Undo", "Redo" та очищення дошки виконують свої завдання.
    - Синхронізація малюнків між учасниками групи відбувається без затримок.
3. Генерація айді та створення кімнати:
    - Кнопка генерації коду створює унікальні айді.
    - Кімната створюється успішно, і хост автоматично переадресовується до створеної кімнати.

Можливості розвитку веб-застосунку

1. Додавання готових карт:

- Це дозволить користувачам обирати попередньо створені шаблони карт для використання під час сесій. Це особливо корисно для тих, хто грає в ігри на зразок Dungeons & Dragons.

2. Нові фігури та інструменти для малювання:
   - Додавання нових фігур (кола, еліпси, полігони) та додаткових інструментів (ластик, заливка кольором) розширить функціонал білої дошки і зробить її більш гнучкою для різних видів малювання.
3. Можливість користувачів взаємодіяти з канвасом при дозволі від хоста:
   - Це додасть новий рівень контролю для хоста, дозволяючи або забороняючи учасникам редагувати білу дошку. Хост зможе керувати ролями учасників та їх доступом до інструментів малювання.
4. Інтеграція з хмарними сервісами:
   - Додавання функції збереження малюнків і сесій на хмарних сервісах дозволить користувачам зберігати свою роботу та продовжувати її пізніше.
5. Покращення функціоналу чату:
   - Додавання можливості надсилати файли та зображення в чат, а також додавання реакцій на повідомлення зробить спілкування більш інтерактивним.

Впровадження цих покращень зробить веб-застосунок ще більш корисним і зручним для користувачів, надаючи їм більше можливостей для творчості та взаємодії.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ТА ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Фізіологічний вплив факторів існування на життєдіяльність людини.

Фізіологічний вплив факторів існування на життєдіяльність людини – це складний процес, що включає взаємодію різних факторів навколишнього середовища, які можуть позитивно або негативно впливати на здоров'я і функціонування людського організму. Ці фактори охоплюють різноманітні аспекти природного та антропогенного середовища, включаючи якість повітря, воду, харчування, кліматичні умови, рівень фізичної активності, стрес та інші. Вплив цих факторів може призводити до змін у фізіологічних процесах, що вимагає відповідних заходів для підтримки оптимального стану здоров'я.

Одним з ключових аспектів є вплив якості повітря на дихальну систему. Забруднення повітря шкідливими речовинами, такими як важкі метали, пил, вуглекислий газ, окиси азоту та інші, може спричиняти респіраторні захворювання, зниження функціональної здатності легенів, алергічні реакції та навіть серцево-судинні захворювання. Регулярне перебування у зонах з високим рівнем забруднення повітря є серйозною загрозою для здоров'я людини і вимагає державних заходів для контролю та зменшення рівня забруднення.

Якість води також відіграє важливу роль у фізіологічному стані людини. Вживання забрудненої води може спричинити різні захворювання, включаючи інфекційні хвороби, токсичні отруєння та хронічні захворювання нирок і печінки. Контроль за якістю питної води та забезпечення доступу до безпечної води є важливим завданням для запобігання негативному впливу на здоров'я.

Харчування є ще одним важливим фактором, що впливає на життєдіяльність людини. Недостатнє або неправильне харчування може призвести до дефіциту необхідних поживних речовин, вітамінів та мінералів, що може викликати різні захворювання, включаючи анемію, ослаблення імунної системи, порушення обміну речовин та інші хронічні захворювання. Водночас, надмірне

вживання їжі, багатої на жири, цукри та солі, сприяє розвитку ожиріння, діабету, серцево-судинних захворювань та інших патологій. Балансоване харчування є основою для підтримання здоров'я та оптимального функціонування організму.

Кліматичні умови також суттєво впливають на фізіологічний стан людини. Екстремальні температури, висока вологість або сухість повітря можуть викликати тепловий стрес, гіпотермію, зневоднення та інші фізіологічні проблеми. Адаптація до кліматичних умов, відповідне вбрання, підтримання оптимального гідратаційного балансу та забезпечення комфортних умов праці та відпочинку є необхідними для збереження здоров'я.

Фізична активність є важливою складовою здорового способу життя. Недостатня фізична активність призводить до зниження функціональних можливостей організму, розвитку гіподинамії, ожиріння, серцево-судинних захворювань, остеопорозу та інших захворювань. Регулярні фізичні навантаження сприяють покращенню роботи серцево-судинної, дихальної, м'язової та інших систем організму, зниженню ризику розвитку хронічних захворювань та підвищенню загального рівня життєдіяльності.

Стрес є невід'ємною частиною сучасного життя і може мати як позитивний, так і негативний вплив на фізіологічний стан людини. Короткочасний стрес може сприяти мобілізації ресурсів організму, підвищенню концентрації уваги та ефективності діяльності. Водночас, тривалий або хронічний стрес може викликати порушення роботи нервової, ендокринної, імунної та інших систем, спричиняючи розвиток психосоматичних захворювань, депресії, тривожних розладів та інших станів. Управління стресом, психоемоційна підтримка, релаксаційні техніки та здоровий спосіб життя є важливими для підтримання психофізіологічного балансу [11].

Висновок: Фактори існування мають різноманітний вплив на фізіологічний стан людини. Забезпечення оптимальних умов для здоров'я та життєдіяльності вимагає комплексного підходу, включаючи контроль за якістю повітря, води та харчування, адаптацію до кліматичних умов, підтримання фізичної активності та

управління стресом. Взаємодія різних факторів середовища та їхній вплив на фізіологічний стан людини потребують постійної уваги та заходів з боку держави та суспільства для забезпечення високого рівня здоров'я та добробуту населення.

4.2 Гігієнічні вимоги до параметрів виробничого середовища приміщень з ВДТ.

Гігієнічні вимоги до параметрів виробничого середовища приміщень з відео-дисплейними терміналами (ВДТ) – це комплекс заходів, спрямованих на забезпечення здорових і безпечних умов праці для працівників, які використовують комп'ютерну техніку у своїй діяльності. Ці заходи включають організаційні, технічні та санітарно-гігієнічні норми, які регламентують параметри виробничого середовища, що впливають на здоров'я та працездатність працівників.

Основне завдання гігієни праці полягає у якісній та кількісній оцінці впливу умов і характеру роботи на організм працівників. На основі цих оцінок розробляються і впроваджуються заходи, що забезпечують максимальну продуктивність праці, мінімізуючи шкідливий вплив факторів трудового процесу на здоров’я. Також, гігієна праці включає розробку та оцінку гігієнічних і лікувально-профілактичних заходів, спрямованих на покращення та збереження здоров’я працівників, а також підвищення їхньої працездатності та продуктивності [12].

Гігієнічні вимоги до виробничого середовища можна поділити на такі як: робочі, санітарно-гігієнічні, психофізіологічні, соціальні, естетичні. Ці пункти впливають на самих працівників, а саме на їх працездатність, та емоційний стан

1. Робочі вимоги.

1.1. Організація робочого місця: Робоче місце повинно бути організоване так, щоб забезпечити максимальну зручність і ефективність виконання завдань. Це включає правильне розташування обладнання, інструментів та матеріалів.

1.2. Ергономіка: Робочі місця повинні відповідати ергономічним стандартам, що включають налаштування стільців, столів, моніторів і клавіатур для зменшення навантаження на м’язи та суглоби.

1.3. Технічне оснащення: Використання сучасного, надійного та безпечного обладнання, що сприяє підвищенню продуктивності праці.

2. Санітарно-гігієнічні вимоги

2.1. Чистота та гігієна: Приміщення повинні регулярно прибиратися, а робочі місця мають бути в чистоті та порядку.

2.2. Вентиляція: Забезпечення ефективної вентиляції для підтримання свіжого та чистого повітря.

2.3. Освітлення: Достатнє природне та штучне освітлення, яке не створює відблисків і не перевантажує зір.

2.4. Мікроклімат: Підтримка оптимальної температури та вологості в робочих приміщеннях.

3. Психофізіологічні вимоги

3.1. Режим праці та відпочинку: Організація робочого часу та перерв для зменшення втоми і напруги. Рекомендуються регулярні перерви для відпочинку очей і м’язів.

3.2. Навчання та адаптація: Психологічна підтримка та адаптація нових працівників до умов праці, регулярне навчання і тренінги для зменшення стресу та підвищення кваліфікації.

3.3. Захист від шуму: Використання засобів захисту від шуму або зменшення рівня шуму в приміщеннях.

4. Соціальні вимоги

4.1. Соціальна підтримка: Забезпечення доступу до соціальних послуг, таких як медична допомога, консультації психологів і соціальних працівників.

4.2. Командний дух: Створення сприятливого соціального середовища, що сприяє командній роботі та підтримці позитивних відносин між працівниками.

4.3. Мотивація: Розробка програм мотивації та заохочення, що сприяють підвищенню продуктивності та задоволеності працею.

5. Естетичні вимоги

5.1. Дизайн приміщення: Створення приємного і естетично привабливого робочого середовища, що включає вибір кольорів, матеріалів і оформлення інтер’єру.

5.2. Озеленення: Використання рослин для поліпшення якості повітря і створення більш приємної атмосфери.

5.3. Зручність та комфорт: Забезпечення комфортних умов для відпочинку та прийому їжі, наявність спеціальних зон відпочинку.

Дотримання гігієнічних вимог до параметрів виробничого середовища приміщень з ВДТ є невід'ємною частиною забезпечення здоров'я та безпеки працівників, що використовують комп'ютерну техніку. Це сприяє підвищенню ефективності роботи та зменшенню ризиків виникнення професійних захворювань.

ВИСНОВКИ

Під час виконання цієї дипломної роботи було проведено дослідження та створено інтерактивну багатокористувацьку карту для гри «Підземелля і дракони» з використанням технологій React та Node.js. Результатом роботи став функціональний додаток, який відповідає вимогам сучасних користувачів та полегшує проведення ігрових сесій. Цей додаток забезпечує інтуїтивно зрозумілий інтерфейс і підтримує інтерактивну взаємодію в режимі реального часу.

Вибір технологій React та Node.js був обумовлений їх високою продуктивністю та можливістю використовувати єдиний стек для клієнтської та серверної частин додатку. Крім того, використання бібліотек, таких як socket.io для реального часу, roughjs для візуалізації графіки з ефектом ручного креслення та react-toastify для сповіщень, дозволило реалізувати ключові функції додатку.

Основною метою роботи було створення зручного інструменту для проведення ігрових сесій, який би поєднував простоту використання з багатим функціоналом. В рамках розробки була створена архітектура додатку, що включає інтерактивну дошку для малювання, систему чатів, управління кімнатами та користувачами. Особлива увага приділялась забезпеченню стабільної роботи та високої продуктивності додатку.

Додаток був успішно протестований, що підтвердило його ефективність та зручність використання. Проведені тести продемонстрували, що додаток відповідає заявленим вимогам та забезпечує стабільну роботу в умовах великої кількості одночасних користувачів.

Таким чином, виконана робота демонструє можливість створення сучасних багатокористувацьких додатків з використанням технологій React та Node.js. Цей проект підкреслює важливість поєднання сучасних технологій, продуманого дизайну та ефективної реалізації для створення високоякісних продуктів.

Подальший розвиток додатку може включати розширення функціональності та інтеграцію нових можливостей на основі зворотного зв'язку від користувачів, що забезпечить ще більшу адаптивність та зручність для гравців.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. React [Електронний ресурс] – Режим доступу до ресурсу: https://react.dev/learn.
2. Node.js [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs.
3. Socket.IO [Електронний ресурс] – Режим доступу до ресурсу: https://socket.io/.
4. Rough.js [Електронний ресурс] – Режим доступу до ресурсу: https://roughjs.com/.
5. React-Toastify [Електронний ресурс] – Режим доступу до ресурсу: https://fkhadra.github.io/react-toastify/introduction.
6. JetBrains WebStorm [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/webstorm.
7. npm [Електронний ресурс] – Режим доступу до ресурсу: https://www.npmjs.com/.
8. Vite [Електронний ресурс] – Режим доступу до ресурсу: https://vitejs.dev/.
9. Yarn [Електронний ресурс] – Режим доступу до ресурсу: https://yarnpkg.com/.
10. Uml star [Електронний ресурс] – Режим доступу до ресурсу: https://staruml.io/.
11. Стиценко Т.Є., Пронюк Г.В., Сердюк Н.М., Хондак І.І. «Безпека життєдіяльності» навчальний посібник / Т. Є.Стиценко, Г. В. Пронюк, Н. М. Сердюк, І. І. Хондак. – Харків: ХНУРЕ, 2018. – 16 – 96 с.
12. Стиценко Т.Є., Пронюк Г.В., Сердюк Н.М., Хондак І.І. «Безпека життєдіяльності» навчальний посібник / Т. Є.Стиценко, Г. В. Пронюк, Н. М. Сердюк, І. І. Хондак. – Харків: ХНУРЕ, 2018. – 115 – 123 с.

ДОДАТКИ

Додаток А

Лістинг коду основних сутностей додатку

Лістинг 1 — код App.jsx

```
import './App.css'
import Forms from "./components/Forms/index.jsx";
import {Routes,Route} from 'react-router-dom'
import {ToastContainer,toast} from 'react-toastify';
import RoomPage from "./pages/RoomPage/index.jsx";
import {useEffect, useState} from "react";

import io from "socket.io-client"

const server = "http://localhost:5000";

const connectionOptions = {
    "force new connection":true,
    reconnectionAttempts: "Infinity",
    timeout:10000,
    transports:["websocket"],

}

const socket = io(server,connectionOptions)

function App  ()  {

    const [user,setUser] = useState(null)
    const [users,setUsers]=useState([])

    useEffect(() => {

        socket.on('userJoined',(data)=>{

            if(data.success){
                console.log('userJoined')
                setUsers(data.users)
            }
            else{
                console.log('userJoined wrong')
            }

        })

        socket.on('allUsers',data=>{
            setUsers(data)
        })

        socket.on('userJoinedMessageBroadcasted',(data)=>{
            toast.info(`${data} joined the room`);
        })
```

```
        socket.on('userLeftMessageBroadcasted',(data)=>{
            toast.info(`${data} left the room`)
        })

    }, []);

    const uuid = () => {
        let S4 = () => {

            return (((1 + Math.random()) * 0x10000) |
0).toString(16).substring(1);

        };

        return (

            S4() + S4() + "-" + S4() + "-" + S4() + "-" + S4() + "-" + S4()
+ S4() + S4()

        );
    };

    return (
    <>
      <div className = "container">
          <ToastContainer />

          <Routes>

              <Route path='/' element={<Forms uuid={uuid} socket={socket}
setUser={setUser}/>}/>

              <Route path="/:roomId" element = {<RoomPage user={user}
socket={socket} users={users}/>}/>

          </Routes>

      </div>
    </>

  )

}

export default App
```

Лістинг 2 — код server.jsx

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

const server = require('http').createServer(app);
```

```
const {Server} = require("socket.io");
const {addUser, getUser, removeUser} = require('./utils/users')

const io = new Server(server) ;

//routes
app.get('/',(req,res)=>{

    res.send('This is mern realtime board sharing app')

})

let roomIdGlobal,imgURLGlobal;

io.on('connection',(socket)=>{

    socket.on('userJoined',(data)=>{

        const {name,userId,roomId,host,presenter} = data;

        roomIdGlobal = roomId

        socket.join(roomId)

        const users =
addUser({name,userId,roomId,host,presenter,socketId:socket.id})

        socket.emit('userJoined',{success:true,users})

        socket.broadcast.emit('userJoinedMessageBroadcasted',name)

        socket.broadcast.to(roomId).emit('allUsers',users)

        socket.broadcast.to(roomId).emit("whiteboardDataResponse",{
            imgURL:imgURLGlobal,
        })

    })

    socket.on("whiteboardData",(data)=>{

        imgURLGlobal = data;

        socket.broadcast.to(roomIdGlobal).emit('whiteboardDataResponse',{
            imgURL:data
        })

    })

    socket.on("message",(data)=>{

        const {message} = data

        const user = getUser(socket.id)
```

```
        removeUser(socket.id)

        if(user){
                        removeUser(socket.id);

            socket.broadcast;


socket.broadcast.to(roomIdGlobal).emit("messageResponse",{message,name:user
.name})
        }

    })

    socket.on('disconnect',()=>{
                const user = getUser(socket.id)

        removeUser(socket.id)

        if(user){

socket.broadcast.to(roomIdGlobal).emit('userLeftMessageBroadcasted',user.na
me)
        }

    })

})

const port = process.env.PORT || 5000;

server.listen(port,()=>{
    console.log('Server is running on http://localhost:5000');
});
```

Лістинг 3 — код App.jsx

```
import {useEffect, useState} from "react";


const Chat = ({setOpenedChatTab,socket})=>{

    const [chat,setChat] = useState([])

    const [message,setMessage] = useState('')

    useEffect(()=>{

        socket.on('messageResponse',(data)=>{

            setChat((prevChats)=>[...prevChats,data])
```

```
        })

    },[])

    const handleSubmit = (e)=>{

        e.preventDefault();

        if(message.trim()!==''){

            setChat((prevChats)=>[...prevChats, {message,name:"You"}])

            socket.emit("message",{message})

            setMessage('')
        }

    }


    return <div className='position-fixed top-0  h-100 text-white bg-dark'
style={{width: "400px", left: "0%"}}>

        <button type='button' className='btn btn-light btn-block w-100 mt-
5'

                onClick={() => setOpenedChatTab(false)}>Close

        </button>

        <div className="w-100 mt-5 p-2 border border-1 border-white
rounded-3" style={{height:"70%"}}>

            {
                chat.map((msg,index)=>(

                    <p key={index*999} className='my-2 text-center w-100
py-2 border border-left-0 border-right-0'>
                        {msg.name}: {msg.message}
                    </p>

                ))
            }

        </div>

        <form onSubmit={handleSubmit} className='w-100 mt-3 d-flex
rounded-3'>

            <input
                type='text'
                placeholder='Enter message'
                className='h-100 border-0 rounded-0 py-2 px-4'
                style={{width:"90%"}} value={message}
                onChange={(e)=>setMessage(e.target.value)}
            />/
```

```
            <button type='submit' className='btn btn-primary rounded-0'>

                Send

            </button>

        </form>

    </div>
}
export default Chat
```

Лістинг 4 — код CreateRoomForm.jsx

```
import {useState} from 'react'
import {useNavigate} from "react-router-dom";

const CreateRoomForm = ({uuid,socket,setUser}) => {

    const [roomId,setRoomId] = useState(uuid())
    const [name,setName]=useState('')

    const navigate = useNavigate()

    const handleCreateRoom=(e)=>{

        e.preventDefault();

        const roomData={
            name,roomId,userId:uuid(),host:true,presenter:true
        }

        setUser(roomData)

        navigate(`/${roomId}`)

        console.log(roomData)

        socket.emit('userJoined',roomData)

    }

    return (

        <form className="form col-md-12 mt-5">

            <div className="form-group">

                <input
                    type="text"
                    value={name}
                    onChange={(e)=>setName(e.target.value)}
                    className="form-control my-2"
                    placeholder="Enter your name"
```

```
                />

            </div>

            <div className="form-group border">

                <div className="input-group d-flex align-items-center
justify-content-center">

                    <input
                        type="text"
                        value={roomId}
                        className="form-control my-2 border-0"
                        placeholder="Generate room code"
                        disabled
                    />

                    <div className="input-group-append ">

                        <button
                            className="btn btn-primary btn-sm me-1"
                            type="button" onClick={()=>setRoomId(uuid)}>
                            Generate
                        </button>

                        <button
                            className="btn btn-outline-danger btn-sm me-2"
                            type="button">
                            Copy
                        </button>

                    </div>

                </div>

            </div>

            <button
                type="submit"
                onClick={handleCreateRoom}
                className="mt-4 btn-primary btn-block form-control">
                Generate Room
            </button>

        </form>
    );

}

export default CreateRoomForm;
```

Лістинг 5 — код App.jsx

```
import './App.css'
import Forms from "./compone
```

Лістинг 6 — код index.jsx

```
import {useState} from "react";
import {useNavigate} from "react-router-dom";

const JoinRoomForm = ({uuid,socket,setUser})=>{

    const [roomId,setRoomId] = useState('');
    const [name,setName] = useState('');

    const navigate = useNavigate();

    const handleRoomJoin= (e)=>{

        e.preventDefault();

        const roomData = {
            name,
            roomId,
            userId:uuid(),
            host:false,
            presenter:false
        };

        setUser(roomData)

        navigate(`/${roomId}`);

        socket.emit('userJoined',roomData)

    }

    return (

        <form className="form col-md-12 mt-5">

            <div className="form-group">

                <input
                    type="text"
                    className="form-control my-2"
                    placeholder="Enter your name"
                    onChange={(e)=> setName(e.target.value)}
                />

            </div>

            <div className="form-group">

                    <input
                        type="text"
                        className="form-control my-2"
                        placeholder="Enter room code"
                        value={roomId}
```

```
                        onChange={(e)=>setRoomId(e.target.value)}
                    />

            </div>

            <button
                type="submit"
                onClick={handleRoomJoin}
                className="mt-4 btn-primary btn-block form-control">Join
Room</button>

        </form>

    )

}

export default JoinRoomForm;
```

Лістинг 7 — код Forms.jsx

```
import "./index.css"
import CreateRoomForm from "./CreateRoomForm/index.jsx";
import JoinRoomForm from "./JoinRoomForm/Index.jsx";

const Forms = ({uuid,socket,setUser})=>{

    return (

        <div className="row h-100 pt-5">

            <div
                className="col-md-4 mt-5 form-box py-3 p-5 border  border-
primary rounded-2 mx-auto mt-5  d-flex flex-column align-items-center">
                <h1 className='text-primary fw-bold'>
                    Create Room
                </h1>

                <CreateRoomForm
                    uuid={uuid}
                    socket={socket}
                    setUser={setUser}
                />

            </div>

            <div className="col-md-4 mt-5 form-box py-3 p-5 border  border-
primary rounded-2 mx-auto mt-5  d-flex flex-column align-items-center">

                <h1 className='text-primary fw-bold'>
                    Join Room
                </h1>

                <JoinRoomForm
                    uuid={uuid}
```

```
                    socket={socket}
                    setUser={setUser}
                />

            </div>

        </div>
    )

}

export default Forms;
```

Лістинг 8 — код WhiteBoard.jsx

```
import { useEffect, useState, useLayoutEffect } from "react";
import rough from "roughjs";

const roughGenerator = rough.generator();

const WhiteBoard = ({ canvasRef, ctxRef, elements,
setElements,tool,color,user,socket }) => {

    const [img,setImg] = useState(null);

    useEffect(()=>{

        socket.on("whiteboardDataResponse",(data)=>{

            setImg(data.imgURL)

        })

    },[])


    if(!user?.presenter) {

        return (

            <div
                className="border border-dark border-3 h-100 w-100
overflow-hidden"
            >
                <img
                    src={img}
                    alt="Real"
                    style={
                    {height:window.innerHeight*2,
                    width:"285%",}
                }
                />

            </div>
        )
```

```
    }
    const [isDrawing, setIsDrawing] = useState(false);

    useEffect(() => {
        const canvas = canvasRef.current;
        canvas.height = window.innerHeight*2;
        canvas.width = window.innerWidth*2;
        const ctx = canvas.getContext("2d");
        ctx.strokeStyle = color;
        ctx.lineWidth = 2;
        ctx.lineCap = 'round';
        ctxRef.current = ctx;
    }, [canvasRef, ctxRef]);


    useEffect(()=>{
        ctxRef.current.strokeStyle = color
    },[color])
    useLayoutEffect(() => {
        if(canvasRef){
            const roughCanvas = rough.canvas(canvasRef.current);
            if(elements.length >0 ){

ctxRef.current.clearRect(0,0,canvasRef.current.width,canvasRef.current.heig
ht)

            }
            elements.forEach((element) => {
                if(element.type==='rect'){
                    roughCanvas.draw(
                        roughGenerator.rectangle(
                            element.offsetX,
```

```
                            element.offsetY,
                            element.width,
                            element.height,
                            {
                                stroke:element.stroke,
                                strokeWidth:5,
                                roughness:0
                            }
                        )
                    )
                }
                else if(element.type==='line'){

roughCanvas.draw(roughGenerator.line(element.offsetX,element.offsetY,elemen
t.width,element.height,{
                            stroke:element.stroke,
                            strokeWidth:5,
                            roughness:0
                        }))
                }

                else if(element.type==='pencil'){
                    roughCanvas.linearPath(element.path,{
                            stroke:element.stroke,
                            strokeWidth:5,
                            roughness:0
                        });
                }
            });
        }

        const canvasImage = canvasRef.current.toDataURL();

        socket.emit('whiteboardData',canvasImage);

    }, [elements, canvasRef, socket]);

    const handleMouseDown = (e) => {

        const { offsetX, offsetY } = e.nativeEvent;

        if(tool==='pencil'){
            setElements((prevElements) => [
                ...prevElements,
                {
                    type: "pencil",
                    offsetX,
                    offsetY,
                    path: [[offsetX, offsetY]],
                    stroke: color,
                }
            ]);
        }
        else if(tool==="line"){
            setElements((prevElements)=>[
                ...prevElements,
```

```
            {
                type:'line',
                offsetX,
                offsetY,
                width:offsetX,
                height:offsetY,
                stroke:'color'
            }
        ])
    }

    else if(tool==='rect'){
        setElements((prevElements)=>[
            ...prevElements,
            {
                type:'rect',
                offsetX,
                offsetY,
                width:0,
                height:0,
                stroke:color
            }
        ])
    }

    setIsDrawing(true);

};

const handleMouseMove = (e) => {

    const { offsetX, offsetY } = e.nativeEvent;

    if (isDrawing) {
        if(tool==='pencil'){
            const { path } = elements[elements.length - 1];

            const newPath = [...path, [offsetX, offsetY]];

            setElements((prevElements) =>
                prevElements.map((ele, index) => {
                    if (index === prevElements.length - 1) {
                        return {
                            ...ele,
                            path: newPath
                        };
                    }else{
                        return ele;
                    }
                })

            );

        }

        else if(tool==="line"){
```

```
                setElements((prevElements)=>
                    prevElements.map((ele, index) => {
                        if(index===elements.length - 1) {
                            return {
                                ...ele,
                                width:offsetX,
                                height:offsetY,
                            };
                        }else{
                            return ele;
                        }
                    })

                )
            }

            else if(tool==='rect'){
                setElements((prevElements)=>
                prevElements.map((ele,index)=>{
                    if(index===elements.length - 1) {
                        return{
                            ...ele,
                            width:offsetX-ele.offsetX,
                            height:offsetY-ele.offsetY,
                        }
                    }else{
                        return ele
                    }
                }))
            }
        }

    };

    const handleMouseUp = () => {

        setIsDrawing(false);

    };



    return (

        <div
            onMouseDown={handleMouseDown}
            onMouseMove={handleMouseMove}
            onMouseUp={handleMouseUp}
            className="border border-dark border-3 h-100 w-100 overflow-
hidden">
            <canvas
                ref={canvasRef}
            />

        </div>
```

```
    );
};

export default WhiteBoard;
```

Лістинг 9 — код RoomPage.jsx

```
import "./index.css"

import {useState,useRef,useEffect} from "react";

import WhiteBoard from "../../components/WhiteBoard/index.jsx";
import Chat from "../../components/ChatBar/index.jsx";

const RoomPage = ({user,socket,users})=>{

    const canvasRef = useRef(null)
    const ctxRef = useRef(null)

    const [tool,setTool]=useState('pencil');
    const [color,setColor]=useState('black');
    const [elements,setElements] = useState([])
    const [history,setHistory]=useState([])
    const [openedUserTab,setOpenedUserTab]= useState(false)
    const [openedChatTab,setOpenedChatTab]= useState(false)



    const handleClearCanvas=()=>{

        const canvas = canvasRef.current;

        const ctx = canvas.getContext('2d')

        ctx.fillRect="white";

        ctx.clearRect(0,0,canvasRef.current.width,canvasRef.current.height)

        setElements([])
    }

    const undo = ()=>{

        setHistory((prevHistory)=>[...prevHistory,elements[elements.length-
1]])

        setElements((prevElements)=>
            prevElements.slice(0,prevElements.length-1)
        )

    }

    const redo = ()=>{
```

```
        setElements((prevElements)=>[
            ...prevElements,history[history.length-1],
        ]);

        setHistory((prevHistory)=>prevHistory.slice(0,prevHistory.length-
1))
    }

    return(

        <div className="row">
            <button
                type='button'
                className='btn btn-dark '
                style={{
                display: 'block',
                position: "absolute",
                top: '5%',
                left: '3%',
                height: '40px',
                width: '100px'
            }}
                onClick={() => setOpenedUserTab(true)}>
                Users
            </button>

            <button
                type='button'
                className='btn btn-primary '
                style={{
                display: 'block',
                position: "absolute",
                top: '5%',
                left: '10%',
                height: '40px',
                width: '100px'
            }}
                onClick={() => setOpenedChatTab(true)}>
                Chats
            </button>

            {
                openedUserTab && (
                    <div
                        className='position-fixed top-0  h-100 text-white
bg-dark'
                        style={{width: "250px", left: "0%"}}>

                        <button

                            type='button'
                            className='btn btn-light btn-block w-100 mt-5'
                            onClick={() => setOpenedUserTab(false)}>Close

                        </button>
```

```
                        <div className="w-100 mt-5 pt-5"></div>

                        {
                            users.map((usr, index) => (
                                <p key={index * 999}
                                   className='my-2 w-100 text-center py-2
'>
                                    {usr.name}{user && user.userId ===
usr.userId && "(You)"}
                                </p>
                            ))
                        }

                    </div>

                )
            }

            {
                openedChatTab && <Chat setOpenedChatTab={setOpenedChatTab}
socket={socket}/>
            }

            <h1
                className="text-center  py-4">"Dungeons and Dragons" custom
map<span
                className="text-primary">[Users
Online:{users.length}]</span>
            </h1>
            {
                user?.presenter && (

                    <div className="col-md-10 mx-auto px-5  mb-3 d-flex
align-items-center justify-content-center">

                        <div className="d-flex col-md-2 justify-content-
center gap-1">

                            <div className="d-flex gap-1 align-items-
center">

                                <label htmlFor="pencil">Pencil</label>

                                <input type="radio" name="tool" id="pencil"
value="pencil" checked={tool == "pencil"}
                                       onChange={(e) => {
                                           setTool(e.target.value)
                                       }}
                                />

                            </div>

                            <div className="d-flex gap-1 align-items-
center">
```

```
                                <label htmlFor="line">Line</label>

                                <input type="radio" name="tool" id="line"
value="line" checked={tool == "line"}
                                       onChange={(e) => {
                                           setTool(e.target.value)
                                       }}
                                />
                            </div>

                            <div className="d-flex gap-1 align-items-
center">

                                <label htmlFor="pencil">Rectangle</label>

                                <input type="radio" name="tool" id="rect"
value="rect" checked={tool == "rect"}
                                       onChange={(e) => {
                                           setTool(e.target.value)
                                       }}
                                />

                            </div>

                        </div>

                        <div className="col-md-3 mx-auto ">

                            <div className="d-flex  align-items-center
justify-content-center ">

                                <label htmlFor="color">Select
Color:</label>

                                <input type="color" id="color"
className="mt-1 ms-3" value={color}
                                       onChange={(e) =>
setColor(e.target.value)}
                                />

                            </div>

                        </div>
                        <div className="col-md-3 d-flex gap-2">

                            <button className="btn btn-primary btn-sm me-1"
disabled={elements.length === 0}
                                    onClick={() => {
                                        undo()
                                    }}
                            >Undo
                            </button>

                            <button className="btn btn-outline-primary mt-
1" disabled={history.length < 1} onClick={() => {
                                redo()
```

```
                            }}>Redo
                            </button>
                        </div>

                        <div className="col-md-2">

                            <button className="btn btn-danger"
                                    onClick={handleClearCanvas}>Clear
Canvas
                            </button>
                        </div>

                    </div>
                )

            }

            <div className="col-md-10 border mx-auto mt-4 canvas-box">

                <WhiteBoard
                    canvasRef={canvasRef}
                    ctxRef={ctxRef}
                    elements={elements}
                    setElements={setElements}
                    tool={tool}
                    color={color}
                    user={user}
                    socket={socket}/>
            </div>

        </div>
    )
}

export default RoomPage
```

Додаток Б – Диск з роботою