Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ВЕБПЛАТФОРМИ ДЛЯ БАГАТОКОРИСТУВАЦЬКОЇ ГРИ З ВИКОРИСТАННЯМ REACT ТА БІБЛІОТЕКИ SOCKET.IO

DEVELOPMENT OF A WEB PLATFORM FOR A MULTIPLAYER GAME USING REACT AND THE SOCKET.IO LIBRARY

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗ-42
Горбач Назар Олегович

Керівник:
Гордєєв Олександр Олександрович

Кваліфікаційну роботу
допущено до захисту
«10» 06 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *бакалавр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«20» 12 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Горбачу Назару Олеговичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *«Розробка вебплатформи для багатокористувацької гри з використанням React та бібліотеки Socket.IO»*
Керівник роботи: *д.т.н., професор Гордєєв. О.О.*
затверджені наказом закладу вищої освіти від *«19» грудня 2024 р.* № *474/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра *«10» червня 2025 р.*
3. Вихідні дані до роботи: *ігровий рушій React, середовище розробки Visual Studio, методичні вказівки до виконання кваліфікаційної роботи бакалавра*
4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *дослідження предметної області, проведення аналізу існуючих аналогів, визначення вимог, проектування проекту та його розробка*
5. Перелік графічного матеріалу: *14 рисунків, 3 таблиці*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Гордєєв О. О.*		
Специфікація вимог до розробленої системи	*Гордєєв О. О.*		
Розробка об'єкта проектування	*Гордєєв О. О.*		
Нормоконтроль	*Вознюк А. В.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	5,72 %		
Академічна доброчесність	*Гордєєв О. О.*		

7. Дата видачі завдання *«20» грудня 2024 р.*

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 04.02.2025 р.	виконано
2	Аналіз проблеми розробки та впровадження об'єкту проектування	до 01.03.2025 р.	виконано
3	Обгрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 15.03.2025 р.	виконано
4	Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних	до 29.03.2025 р.	виконано
5	Практична реалізація об'єкта проектування та розробка бази даних	до 26.04.2025 р.	виконано
6	Тестування та налагодження об'єкта проектування	до 03.05.2025 р.	виконано
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру	до 10.06.2025 р.	виконано

Здобувач вищої освіти _____ Назар ГОРБАЧ

Керівник кваліфікаційної роботи _____ Олександр ГОРДЄЄВ

АНОТАЦІЯ

Горбач Н. О. Розробка вебплатформи для багатокористувацької гри з використанням React та бібліотеки Socket.IO. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків та списку використаних джерел.

У першому розділі здійснено аналіз предметної області, порівняння існуючих аналогів продукту та технологій, сформульовано завдання на кваліфікаційну роботу бакалавра. В другому розділі було сформульовано функціональні та нефункціональні вимоги до проекту, описано проектування продукту, проаналізовано середовища розробки та програмне забезпечення, виконаний їх вибір для виконання поставленого завдання. В третьому розділі здійснено опис процесу розробки проекту та проведення його тестування. У висновках описано результати виконання кваліфікаційної роботи.

Ключові слова: ігровий застосунок, ігровий рушій, ігрові механіки, користувач, клас, прототип, інструмент розробки, скрипт, система.

ABSTRACT

Horbach N. Development of a Web Platform for a Multiplayer Game Using React and the Socket.IO Library. Bachelor's Qualification Work in the Educational Program "Software Engineering", Specialty "Software Engineering". Lutsk National Technical University. Lutsk, 2025.

The bachelor's qualification thesis consists of an introduction, three chapters, conclusions and a list of references.

The first section analyzes the subject area, compares existing analogues of the product and technologies, formulates tasks for the bachelor's qualification work. The second section formulates functional and non-functional requirements for the project, describes the product design, analyzes the development of the environment and software, and selects them to perform the task. The third section describes the process of developing the project and conducting its testing. The conclusions describe the results of the qualification work.

Keywords: game application, game engine, game mechanics, user, class, prototype, development tool, script, system.

ЗМІСТ

ВСТУП

Актуальність теми. За останнє десятиліття у світі виник і сформувався новий напрям – комп'ютерні ігри. Сьогодні ринок розваг є одним з найбільш прибуткових. У той момент, коли почалася інформаційно-технологічна революція, світ стрімко розвивався, створюються все більш досконалі комп'ютерні системи, щоб полегшити життя людині, так як і час відпочинку.

Комп'ютерні розваги роблять життя людей все багатшим і багатшим. В результаті, цей потужний сектор економіки приносить величезні доходи, неймовірні емоції. Тому не випадково, що особлива роль у сучасному житті присвячується комп'ютерним іграм.

Одним із прикладів комп'ютерних ігор це браузерні ігри. Браузерна гра – комп'ютерна гра, що використовує браузерний інтерфейс і зазвичай не потребує установки на комп'ютер додаткових додатків, крім самого браузера та іноді плагіна для нього. Браузерні ігри можна розділити на однокористувацькі – розраховані на одного користувача і багатокористувацькі – розраховані на двох і більше користувачів, що було реалізовано у кваліфікаційній роботі.

Дана кваліфікаційна робота присвячена вирішенню актуального практичного завдання, що полягає у проектуванні та розробці веб-платформи багатокористувацької гри між собою під час гри з допомогою чату та відеочату. Актуальність кваліфікаційної роботи полягає в тому, що веб-платформа дає змогу швидко розпочати вибрану гру з опонентом та комунікувати з допомогою відеочату.

Метою кваліфікаційної роботи є створення вебплатформи для багатокористувацької гри на базі сучасного фронтенд-фреймворку React та бібліотеки Socket.IO для забезпечення реального часу взаємодії між гравцями. Проєкт фокусується на реалізації гри Chess (шахи) як основної, з можливістю подальшого розширення на інші ігри (наприклад, Checkers або Tic-Tac-Toe), відповідно до представленого візуального інтерфейсу вибору гри.

Об’єктом кваліфікаційної роботи є процес розробки вебплатформи для багатокористувацької гри з використанням React та бібліотеки Socket.IO.

Предметом кваліфікаційної роботи є методи, інструменти та технології створення інтерактивних вебзастосунків, зокрема у сфері розробки багатокористувацьких ігор із використанням React і Socket.IO.

Для досягнення цілей поставленої мети, потрібно виконати такий перелік завдань:

– проаналізувати наявні онлайн-рішення для багатокористувацьких шахових ігор, зокрема за критеріями технологій, дизайну та зручності використання;

– дослідити можливості React для реалізації динамічного UI з інтерактивними елементами, такими як вибір гри, введення Room ID та з’єднання з іншими гравцями;

– розробити серверну логіку на Node.js з використанням Socket.IO для обміну подіями між клієнтами в режимі реального часу (створення кімнат, хід гравця, завершення гри);

– створити адаптивний і привабливий інтерфейс користувача, відповідно до сучасних стандартів UI/UX, з урахуванням легкості навігації та простоти входу в гру;

– реалізувати основні правила гри в шахи, шашки, включаючи логіку пересування фігур, чергування ходів;

– забезпечити можливість підключення до гри через унікальний Room ID, що дає змогу організовувати приватні або публічні ігрові сесії;

– провести тестування розробленої платформи, включаючи перевірку стабільності з’єднання, коректності ігрової логіки та адаптивності інтерфейсу на різних пристроях.

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧИХ ВЕБ-ПЛАТФОРМ БАГАТОКОРИСТУВАЦЬКОЇ ГРИ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Поняття та складові веб-платформи

Веб платформа – це сукупність технологій що розроблені як відкриті стандарти Консорціумом Всесвітнього павутиння (W3C) та іншими органами стандартизації: WHATWG, Unicode Consortium, IETF і Ecma International. Це гіперонім, що був введений W3C, і в 2011 році він був визначений CEO W3C Джеффом Яффе як «платформа для інновацій, консолідації та економічної ефективності». Створення стандартів з урахуванням The evergreen Web (де мають місце швидкість, автоматизоване оновлення ПЗ, співпраця вендорів, стандартизація і конкуренція) дозволило додати нові можливості при вирішенні ризиків щодо безпеки та конфіденційності.

Веб платформа включає в себе технології – комп'ютерні мови та API, які спочатку були створені для публікації веб сторінок. До неї входять HTML, CSS 2.1, CSS, SVG, MathML, WAI-ARIA, ECMAScript, WebGL, Web Storage, Indexed Database API, Web Components, WebAssembly, Web Workers, WebSocket, Geolocation API, Server-Sent Events, DOM Events, Media Fragments, XMLHttpRequest, Cross-Origin Resource Sharing (CORS), File API, RDFa, WOFF, HTTP, TLS 1.2, and IRI [1].

Найбільш поширені веб-платформ виділяють:

- пошукові системи;
- веб-хостинги;
- веб-пошта;
- веб-архіви.

1.1.1 Огляд сучасних веб-платформ для багатокористувацької гри

На сьогодні існує велика кількість веб-платформ, що забезпечують доступ до багатокористувацьких ігор із різноманітним функціоналом та ігровим контентом. Кожна з них має свої особливості, переваги й недоліки, які варто

враховувати при виборі платформи для реалізації або використання ігрового продукту. Залежно від цільової аудиторії, технічних можливостей, рівня інтерактивності та підтримуваних технологій, платформи відрізняються за підходами до побудови ігрового процесу, масштабованістю, продуктивністю та рівнем безпеки. У цьому розділі буде проведено огляд найпопулярніших веб-платформ для багатокористувацьких ігор, з метою визначення їх доцільності використання при розробці власного ігрового проєкту.

1.1.2 Порівняльний аналіз комерційно успішних аналогів

Порівняльний аналіз успішних комерційних проектів таких як Lichess.org та Chess.com, дозволяє виявити певні підходи до розробки, що забезпечують їх успіх на ринку:

- застосування технік динамічної та якісної обробки ресурсів та графіки;
- адаптація технічних рішень під різну специфіку ігрових продуктів;
- гравець повинен отримати новий та унікальний для себе досвід.

Chess.com застосовує бізнесову модель freemium. Головні можливості сайту безкоштовні, але гравці мають платити за додаткові можливості.

Відвідувачі можуть грати в живому режимі і режимі заочних шахів. Гравці можуть також грати проти шахових рушіїв, а також брати участь у матчах за голосуванням, в яких гравці утворюють команди й голосують за найкращі ходи. Додаткові можливості включають: програмовані тренери з тактики, шахові форуми, статті, новини, завантаження, дебютні бази даних, групи, трансляції наживо, щоденні шахові задачі, тренери онлайн і базу даних шахових партій з більш ніж 2 мільйонами ігор.

Компанія публікує велику кількість статей на різні теми пов’язані з шахами, включаючи шахову стратегію, теорію дебютів та історію. Серед постійних дописувачів Грегорі Серпер, Брюс Пандольфини, Рафаель Лейтао, Ден Хейсман, Джеремі Сільман, Петар Генов і Наталія Погоніна.

Chess.com застосовує політику проти використання шахових рушіїв у всіх видах гри, окрім не рейтингових ігор, де обидва гравці домовились про це. Сайт

прийняв політику «name and shame» проти нечесної гри у 2009 році і веде чорний список.

Основна функція сайту Lichess – грати в шахи онлайн або спілкуватися з іншими гравцями за допомогою різних параметрів керування часом. Усі ігри, в які грають зареєстровані користувачі, зберігаються на сервері зі статистикою та можливостями пошуку [2]. Гра може бути як грою, яка не впливає на рейтинги гравців, так і рейтингом (вибирається користувачем), який використовується для обчислення рейтингів гравців, розбитих за контролем часу (бліц, класичні шахи тощо). Шаховий двигун Stockfish можна використовувати для аналізу всіх ігор. Окрім живих ігор, користувачі також можуть грати проти комп'ютерів [3].

Lichess надає широкі можливості для вдосконалення тактичних компонентів гри (вирішення шахових задач, пошук матів, пошук найкращого ходу), покращення володіння шаховими нотаціями та підготовка знань для дебютів. На додаток до різних варіантів контролю часу в стандартних шахах, сайт підтримує наступні варіанти:

- шахи-960;
- цар гори;
- шахові піддавки (анти-шахи);
- атомні шахи;
- три Шахові шахи.

1.1.3 Графіка та анімації

У представленому застосунку реалізована мінімалістична 2D-графіка, яка повністю відповідає логіці гри в шахи та підтримує її стратегічну природу. Вибір лаконічного дизайну був зумовлений потребою забезпечити чітку візуальну ідентифікацію елементів шахової дошки та фігур без зайвого візуального шуму. Завдяки цьому інтерфейс не перевантажений деталями, що дозволяє користувачеві зосередитись безпосередньо на геймплейному процесі.

Основними елементами графіки є спрайти шахових фігур, які мають фіксоване положення та не потребують складної анімації. Візуальне оформлення шахівниці виконано у контрастній зелено-білій палітрі, що є загальноприйнятим

стандартом в цифрових шахових інтерфейсах. Така кольорова гама забезпечує легке сприйняття клітин та зменшує візуальне навантаження на очі гравця.

Анімація в грі реалізована на базовому рівні – переважно через візуальні переходи при пересуванні фігур. Завдяки використанню сучасних засобів рендерингу у React, таких як CSS transitions або canvas-анімація, досягається плавний та зрозумілий рух, який моделює логіку гри без надмірного навантаження на систему [4].

Інтерфейс гри також містить супутні візуальні компоненти: поле чату, інтерактивну кнопку запуску відеочату, та текстові підказки для користувача. Усі елементи витримані в єдиному стилі з використанням стандартної типографіки та відтінків, що гармонійно поєднуються з ігровим полем. Таким чином, графічне рішення у проєкті не лише виконує естетичну роль, а й підтримує функціональність, зручність та доступність користування застосунком.

1.1.4 Алгоритми пошуку шляху

У розробці гри Chess, яка реалізована в мультимодальному інтерфейсі з можливістю вибору гри (Chess, Checkers, Tic-Tac-Toe) та підключенням до кімнати за допомогою Room ID, важливим компонентом є система навігації та позиціонування фігур. Незважаючи на те, що шахи є покроковою грою без потреби в реальному русі вільного об’єкта по сцені, алгоритми пошуку шляху застосовуються для:

- визначення валідності ходу;
- розрахунку допустимих траєкторій фігур (особливо для складних випадків, як у ферзя або коня);
- імітації анімаційних переходів фігур по клітинках;
- підтримки AI-бота для аналізу ходів (в майбутньому розширенні гри).

Для реалізації таких механік було розглянуто декілька підходів до побудови логіки навігації.

NavMesh зазвичай використовується у 3D-іграх, його принципи можна адаптувати під 2D-середовище шахової дошки. Сітка NavMesh моделює прохідні

зони (в даному випадку – доступні клітинки для кожної фігури). Основні переваги:

– швидке оновлення маршруту після кожного ходу;

– підтримка анімацій переходу фігур;

– можливість враховувати перепони (інші фігури, наприклад);

– адаптація до динамічних змін (наприклад, коли фігура зникає після взяття).

Недоліки: надмірність для простої шахової логіки, складність у розрахунку для точних ходів, неефективність для вузьких сіток, як шахова.

Waypoint Graphs базується на фіксованих точках (у випадку шахів – центри клітинок), що з'єднані між собою логічними зв'язками. Його перевага є простота реалізації:

– кожна клітинка – окремий вузол графу;

– шлях – це просто перевірка наявності зв'язку між поточною позицією та цільовою;

– ідеально підходить для симуляції ходів та базової AI-логіки;

– однак даний метод не адаптується до змін динамічно без написання додаткової логіки (наприклад, наявність фігури, що блокує хід).

FlowFields частіше використовується для масового руху (наприклад, у стратегіях), його можна використати для AI-планування ходів. Застосування цього методу дозволяє:

– візуалізувати «тиск» ходів від AI-гравця;

– будувати траєкторії впливу фігур на поле (наприклад, контрольовані клітинки);

– підтримувати складні стратегії оцінки поля.

Недоліки – складність реалізації, високі витрати ресурсів, та надмірність для покрокової гри.

У результаті, провівши дослідження даних систем навігації було створено порівняльну таблицю, щоб детально розглянути їхні відмінності (табл. 1.1).

Таблиця 1.1 – Оцінка ефективності навігаційних алгоритмів для шахової гри

Критерій	NavMesh	Waypoints	FlowFields
Швидкість	Висока	Висока	Низька
Придатність до шахів	Середня	Висока	Середня
Реакція на зміни	Висока	Низька	Висока
Складність реалізації	Висока	Низька	Висока
Візуалізація ходів	Так	Обмежено	Так

Для реалізації гри Chess в запропонованому мінімалістичному інтерфейсі (вибір гри, Room ID, логічна навігація), найдоцільнішим є використання Waypoint Graphs, оскільки вони відповідають сутності самої гри (обмежена кількість клітинок, статична сцена, простота перевірки ходів). Однак для візуального ефекту та можливості реалізації AI-механік у майбутньому можуть бути використані елементи FlowFields або адаптований NavMesh.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Метою кваліфікаційної роботи є створення вебплатформи для багатокористувацької гри на базі сучасного фронтенд-фреймворку React та бібліотеки Socket.IO для забезпечення реального часу взаємодії між гравцями. Проєкт фокусується на реалізації ігор Chess, Checkers, Tic-Tac-Toe, відповідно до представленого візуального інтерфейсу вибору гри.

Для виконання цього завдання, потрібно виконати наступні ключові етапи:

– проаналізувати наявні онлайн-рішення для багатокористувацьких шахових ігор, зокрема за критеріями технологій, дизайну та зручності використання;

– дослідити можливості React для реалізації динамічного UI з інтерактивними елементами, такими як вибір гри, введення Room ID та з’єднання з іншими гравцями;

– розробити серверну логіку на Node.js з використанням Socket.IO для обміну подіями між клієнтами в режимі реального часу (створення кімнат, хід гравця, завершення гри);

– створити адаптивний і привабливий інтерфейс користувача, відповідно до сучасних стандартів UI/UX, з урахуванням легкості навігації та простоти входу в гру;

– реалізувати основні правила гри в шахи, шашки, включаючи логіку пересування фігур, чергування ходів;

– забезпечити можливість підключення до гри через унікальний Room ID, що дає змогу організовувати приватні або публічні ігрові сесії;

– провести тестування розробленої платформи, включаючи перевірку стабільності з’єднання, коректності ігрової логіки та адаптивності інтерфейсу на різних пристроях.

У результаті буде створено робочий прототип вебплатформи для гри в шахи з можливістю розширення, що забезпечує зручну, швидку та стабільну взаємодію між гравцями в реальному часі за допомогою сучасних веб-технологій.

Висновки до розділу 1

Проведений аналіз сучасного стану розробки ігрових продуктів дозволив виявити особливості технічних та дизайнерських рішень, пов’язані з розробкою ігор. На основі дослідження аналогів проекту, особливостей застосування технологій та можливостей ігрових рушіїв було сформовано чіткий перелік завдань для майбутньої реалізації проекту. Показано доцільність використання конкретних технологій, таких як React для побудови інтерфейсу користувача та Socket.IO для організації взаємодії в реальному часі.

Результати аналізу стали підґрунтям для обґрунтованого вибору інструментів та підходів, які сприятимуть створенню ефективної, доступної та функціональної вебплатформи для багатокористувацької гри.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблювального програмного забезпечення та проектування програмного забезпечення

У сучасних умовах розвитку веб-технологій багатокористувацькі ігри, що працюють без встановлення додаткового програмного забезпечення, стають все більш популярними. Гравці очікують швидкий доступ, мінімальні затримки під час сеансів гри та можливість взаємодії з іншими користувачами в реальному часі. Застосування бібліотеки Socket.IO дозволяє реалізувати обмін даними між клієнтом і сервером у режимі реального часу, що є ключовим для багатокористувацького геймплею. Інтерфейсна частина платформи реалізується з використанням React – однієї з найпопулярніших бібліотек для створення динамічних вебзастосунків.

Платформа орієнтована на прості за механікою, але конкурентні за ігровим процесом ігри (наприклад, шахи, шашки, хрестики-нулики), де важлива синхронність дій між користувачами.

Функціональні вимоги:

– реєстрація та автентифікація користувачів. Користувач повинен мати змогу створити обліковий запис або увійти в систему. Повинна підтримуватися система ідентифікації через унікальне ID кімнати;

– створення ігрової кімнати;

– можливість створення унікального сеансу гри з автоматичним або вручну заданим кодом;

– приєднання іншої сторони через ID кімнати;

– синхронізація гри в реальному часі;

– дії одного гравця миттєво відображаються в інтерфейсі іншого;

– запобігання одночасним ходам (черговість);

– гнучка архітектура гри;

– платформа повинна мати можливість підключати різні ігри (наприклад, шахи, шашки) через єдиний фреймворк;

– відображення результатів гри;

– виведення повідомлень про завершення гри.

Нефункціональні вимоги:

– забезпечення кросплатформенності вебплатформи, зокрема її коректної роботи в сучасних браузерах на персональних комп’ютерах, планшетах та смартфонах;

– забезпечення високої продуктивності платформи, зокрема мінімізації затримки передачі даних, яка не повинна перевищувати 100 мс;

– забезпечення безпеки платформи шляхом захисту сеансів гри та персональних даних користувачів із використанням протоколів WebSocket (із захищеним з’єднанням WSS) та HTTPS;

– підтримка масштабованості платформи з можливістю розширення функціоналу шляхом додавання нових ігор без перебудови основної архітектури.

На рисунку 2.1 представлено діаграму варіантів використання користувачем функціоналу системи.

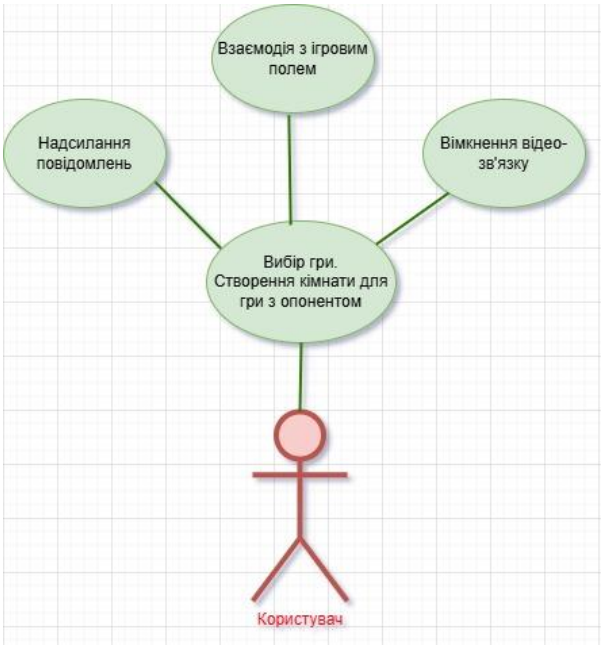

Рисунок 2.1 – Діаграма варіантів використання

Додатково було створено діаграму активності зображено на рисунку 2.2 яка демонструє послідовність дій і прийняття рішень у грі, що є важливим для розуміння логіки платформи та реалізації її в коді.

Рисунок 2.2 – Діаграма активності

Діаграма станів в шахах у багатокористувацькому середовищі та переходи між ними в процесі взаємодії користувача з платформою. Вона є важливою частиною моделювання логіки гри, оскільки дозволяє візуалізувати, як змінюється стан системи залежно від дій користувачів та ігрових подій.

Основні стани, які представлено на діаграмі:

– вхід до системи – початковий етап, на якому користувач проходить процедуру авторизації на платформі;

– очікування супротивника – стан, у якому один з гравців створив кімнату та очікує підключення іншого;

– гра – активний стан, під час якого відбувається взаємодія гравців у режимі реального часу;

– перемога / поразка – завершальний стан гри, залежно від результату партії;

– перезапуск гри (або інша дія після завершення) – дає можливість гравцям повторити гру або повернутись до головного меню.

Даний рисунок 2.3 дозволяє структурувати життєвий цикл сесії шахової гри та формалізувати логіку переходів між станами. Вона є корисною як на етапі проєктування, так і при тестуванні поведінки програми.

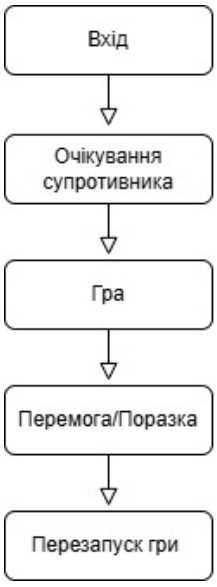

Рисунок 2.3 – Діаграма станів

Першим прототипом інтерфейсу, яке було розроблено, стало головне меню гри. Головне меню зображене на рисунку 2.4 повинно містити вибір ігор, поле вводу номера кімнати та одну кнопки приєднатися.

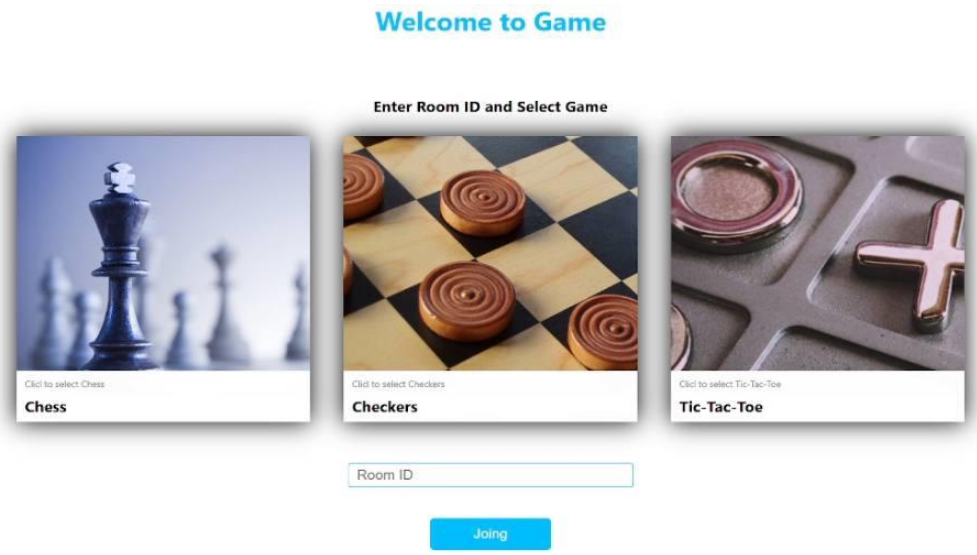

Рисунок 2.4 – Дизайн головного меню

Наступним чином, було створено прототип спливаючої підказка, де розміщена інформація про гру, зображено на рисунку 2.5.

Рисунок 2.5 – Дизайн спливаючої підзказка

Саме ці вспливаючі підказки є індивідуальні на кожній грі, а також з’являються коли обираєш гру.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Сучасні методи створення веб-застосунків передбачають велику кількість підходів до реалізації інтерактивних, масштабованих та динамічних рішень, особливо коли йдеться про багатокористувацькі платформи, що працюють у реальному часі. З урахуванням зростання вимог до швидкодії, стабільності та адаптивності, особливу увагу необхідно приділити вибору засобів розробки. Для вирішення поставленого завдання, а саме – розробки вебплатформи для гри в шахи між кількома користувачами, було проаналізовано сучасні бібліотеки та середовища веб-розробки. Основними критеріями вибору стали: підтримка роботи в реальному часі, продуктивність, кросплатформеність, легкість масштабування та підтримка інтерактивного інтерфейсу.

У ході порівняльного аналізу були розглянуті такі технології: React, Angular, Vue.js – для побудови інтерфейсної частини; Socket.IO, WebRTC, Firebase – для реалізації обміну даними між клієнтом і сервером. Також проаналізовано різні підходи до реалізації серверної логіки на Node.js, Python та Golang. Вибір зупинився на React як основному фреймворку для створення клієнтського інтерфейсу, та Socket.IO як інструменті для реалізації обміну даними між користувачами в режимі реального часу. Серверна частина була реалізована за допомогою Node.js, що дозволяє досягнути високої швидкодії при обробці численних одночасних з’єднань [4].

React – це сучасна JavaScript-бібліотека, що надає можливість побудови компонентного інтерфейсу, який легко адаптується до вимог гри [4]. Завдяки використанню віртуального DOM, забезпечується швидка реакція інтерфейсу на події гравців. Бібліотека підтримує використання хуків та локальних станів компонентів, що дозволяє ефективно реалізувати такі елементи, як шахова дошка, чат, індикатори стану гри, поле вводу імені чи Room ID. Компонентна модель React дозволила модульно побудувати всю платформу, розділивши її на незалежні частини з чіткими обов’язками та логікою.

Socket.IO виступив основним засобом для реалізації передачі даних у реальному часі. За його допомогою була реалізована система створення та підключення до кімнат за Room ID, передача інформації про хід гравця, обробка подій початку гри, завершення або її скидання. Перевагою використання саме цієї бібліотеки є її подієва модель: замість опитування серверу, обидва клієнти реагують на події одразу, що дозволяє досягти практично миттєвої синхронізації дій. Socket.IO також має підтримку повторного з'єднання, перевірки активності з'єднання та масштабування через socket.io-redis.

Серверна частина платформи реалізована з використанням Node.js – асинхронного середовища виконання JavaScript, яке чудово підходить для обробки одночасних з'єднань без блокування потоків. Використання Express.js як надбудови над Node.js дозволило організувати структуру маршрутизації та обробки запитів, а вбудовані можливості Node.js у роботі з подіями ідеально поєднуються з логікою Socket.IO. Така архітектура дозволяє обробляти десятки одночасних ігор без втрат у продуктивності. Для реалізації шахової логіки була використана бібліотека chess.js, яка забезпечує валідацію ходів, визначення шаху, мату, нічиї, а також історію партій [5].

Окрім цього, було розглянуто кілька альтернативних рішень. Так, Vue.js виявився простим у використанні, проте має менш розвинену екосистему для складних SPA-додатків. Angular відзначається високою потужністю, однак потребує більше часу на налаштування, тому для індивідуальної розробки не був обраний. У свою чергу, WebRTC показав чудову швидкодію, але є надлишковим у задачах, де передається лише інформація про ходи. Firebase забезпечує зручну синхронізацію, але не надає гнучкої обробки подій як Socket.IO. Тому вибраний стек технологій – React + Socket.IO + Node.js – виявився найбільш оптимальним для реалізації поставленого завдання [6].

Наступним чином, було сформовано порівняльну таблицю (табл. 2.1) наведено аналіз деяких основних засобів, розглянутих під час дослідження.

Таблиця 2.1 – Порівняльна таблиця технологій для реалізації веб-платформи

Критерії	React	Vue.js	Angular	Socket.IO
Компонентна модель	Так	Так	Так	-
Крива навчання	Середня	Низька	Висока	Низька
Підтримка спільноти	Висока	Середня	Висока	Висока
Гнучкість налаштування	Висока	Висока	Середня	Висока
Робота в реальному часі	Через Socket	Через Socket	Через RxJS	Так
Подієва модель взаємодії	Так	Так	Так	Так

У підсумку, для реалізації кваліфікаційної роботи було обрано стек технологій React + Socket.IO + Node.js як найбільш збалансоване рішення, що забезпечує гнучкість у реалізації, підтримку передачі даних у реальному часі, зручність побудови інтерфейсу та високий рівень продуктивності. У наступному розділі буде розглянуто специфікацію функціональних вимог до системи, її архітектуру та ключові компоненти реалізації.

Розробка сучасних вебплатформ потребує використання ефективних і надійних середовищ програмування, що підтримують сучасні вебтехнології, мають гнучкий інтерфейс, розширення, автодоповнення та інструменти для контролю версій. У зв’язку з цим, на етапі підготовки до реалізації багатокористувацької гри в шахи з використанням React та Socket.IO було здійснено аналіз кількох інтегрованих середовищ розробки, зокрема: Visual Studio Code, WebStorm і Sublime Text.

Особливо варто відзначити інтеграцію з Git, зручний термінал прямо в інтерфейсі та підтримку дебагінгу Node.js, що дозволило ефективно реалізувати серверну логіку гри. Завдяки підтримці WebSocket-розширень та емуляції запитів, можна тестувати логіку Socket.IO без потреби запуску зовнішніх сервісів. Простота конфігурації, швидкодія, а також безкоштовна ліцензія зробили Visual Studio Code оптимальним вибором для реалізації проекту [7].

WebStorm – це комерційне середовище від JetBrains, що спеціалізується на JavaScript-розробці. Воно має глибоку інтеграцію з фреймворками React, Vue, Angular, а також з Node.js. Відмінною рисою WebStorm є надзвичайно потужні інструменти аналізу коду, автоматичного виправлення помилок, підсвітки залежностей та інтелектуального автодоповнення. Крім того, він підтримує Docker, Git, CI/CD-пайплайни та засоби тестування (Jest, Mocha).

Хоча WebStorm забезпечує комфортну роботу з великими JavaScript-проєктами та має зручний UI, для невеликих або навчальних проектів його ліцензійна модель (оплата за підпискою) може стати перешкодою [8]. Також для його використання потрібен комп'ютер із достатньою кількістю оперативної пам'яті – у протилежному випадку можливе уповільнення при відкритті великих репозиторіїв.

Sublime Text – це легке текстове середовище, яке підтримує синтаксис JavaScript, HTML, CSS, але не має вбудованих функцій для налагодження або повної інтеграції з екосистемою Node.js. [8]. Завдяки своїй швидкодії та мінімалізму, Sublime часто використовується для швидкої правки або аналізу фрагментів коду, однак для повноцінної розробки фронтенду та серверної логіки вебплатформи він є недостатнім.

Наступним чином, було сформовано порівняльну таблицю (табл. 2.2) середовищ розробки, щоб розглянути їхні відмінності та обрати її для виконання кваліфікаційної роботи.

Таблиця 2.2 – Порівняльна таблиця середовищ розробки

Критерій	Visual Studio Code	WebStorm	Sublime Text
Ліцензія	Безкоштовна	Платна	Умовно безкоштовна
Підтримка React/Node.js	Повна	Повна	Часткова
Продуктивність	Висока	Висока	Дуже висока
Підтримка розширень	Дуже велика	Вбудована	Package Control
Інтеграція з Git	Так	Так	Так
Наявність дебагера	Так (через плагіни)	Так (вбудований)	Ні

У результаті аналізу було обрано Visual Studio Code як основне середовище розробки проєкту. Програма дозволяє ефективну розробку клієнтської і серверної частин, реалізувати інтеграцію з Git, провести тестування Socket.IO-підключення. Безкоштовна ліцензія, багатофункціональність і розширюваність зробили його найкращим вибором для реалізації вебплатформи гри в шахи у межах кваліфікаційної роботи.

Висновки до розділу 2

У другому розділі, було визначено функціональні та нефункціональні вимоги до проекту. Проведено опис систем притаманним гравцю та ворогам. Створено прототип інтерфейсу ігрового застосунку. Проаналізовано ігрові рушії та середовища розробки, створено порівняльні таблиці для вибору програмного забезпечення.

РОЗДІЛ 3
ВЕБПЛАТФОРМИ ДЛЯ БАГАТОКОРИСТУВАЦЬКОЇ ГРИ З ВИКОРИСТАННЯМ REACT ТА БІБЛІОТЕКИ SOCKET.IO

3.1 Практична реалізація об’єкта проектування

Веб-платформа, що реалізується в межах цього проєкту, має двокомпонентну архітектуру, яка включає:

– серверну частину, що відповідає за обробку запитів, обмін даними між користувачами в реальному часі та управління ігровою логікою;

– клієнтську частину, тобто безпосередньо веб-інтерфейс, з яким взаємодіє користувач під час гри.

Такий розподіл дозволяє досягти гнучкості, масштабованості та ефективного розподілу ресурсів між клієнтською та серверною логікою гри. У цьому розділі детально описано процес реалізації обох складових платформи.

На рисунку 3.1 показана файлова структура серверу.

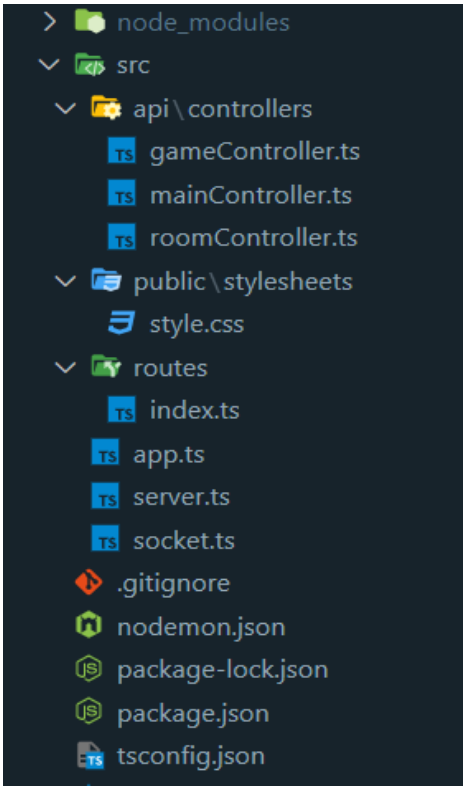

Рисунок 3.1 – Файлова структура серверу

Розгортання проекту реалізовував в середовище розробки – Visual Studio Code розпочав зі створення папок для розповсюдження та зручності використання компонентів.

Створимо файл «package.json», який буде описувати створений проект. Для його створення використав вбудований термінал у середовищі розробки. Цей файл містить назву проекту, версію, скрипти та пов’язані бібліотеки (ліст. 3.1).

Лістинг 3.1 – Налаштування серверу

```
{
  "name": "socketio-server",
  "version": "0.0.0",
  "private": true,
  "license": "MIT",
  "scripts": {
    "start": "nodemon"
  },
  "dependencies": {
    "cookie-parser": "~1.4.4",
    "cors": "^2.8.5",
    "debug": "~2.6.9",
    "express": "~4.16.1",
    "glob": "^7.1.7",
    "http-errors": "~1.6.3",
    "jade": "~1.11.0",
    "morgan": "~1.9.1",
    "reflect-metadata": "^0.1.13",
    "socket-controllers": "^0.5",
    "socket.io": "^4.1.2"
  },
  "devDependencies": {
    "@types/node": "^16.0.0",
    "nodemon": "^2.0.9",
    "ts-node": "^10.0.0",
    "typescript": "^4.3.5"
  }
}
```

кінець лістингу 3.1

Далі запустимо локальний сервер для розгортання проекту. Для цього створимо файл «server.ts» та проведемо ініціалізацію. Сервер розгортаємо на проті 9000. Підключення сервера зображено в лістингу 3.2.

Лістинг 3.2 – Підключення сервера

```
import "reflect-metadata";
import app from "./app";
var debug = require("debug")("socketio-server:server");
import * as http from "http";
import socketServer from "./socket";

var port = normalizePort(process.env.PORT || "9000");
app.set("port", port);

var server = http.createServer(app);

server.listen(port);
server.on("error", onError);
server.on("listening", onListening);
```

кінець лістингу 3.2

Створюємо файл «mainController.ts». В даному файлі реалізовуємо створення Socket користувача показано в лістингу 3.3.

Лістинг 3.3 – Створення socket

```
public class Sword : MonoBehaviour
import {
  ConnectedSocket,
  OnConnect,
  SocketController,
  SocketIO,
} from "socket-controllers";
import { Socket, Server } from "socket.io";
@SocketController()
export class MainController {
  @OnConnect()
  public onConnection(
    @ConnectedSocket() socket: Socket,
    @SocketIO() io: Server
  ) {
    console.log("New Socket connected: ", socket.id);
    socket.on("custom_event", (data: any) => {
      console.log("Data: ", data);
    });
  }
}
```

кінець лістингу 3.3

У файлах «roomController.ts» та «gameController.ts» реалізовуємо взаємодію між користувачами під час гри, створення та підключення до однієї кімнати. Функції роботи з кімнатами та взаємодії із гравцями зображено в лістингу 3.4.

Лістинг 3.4 – Взаємодія між гравцями

```
public class Sword : MonoBehaviour
{
    @SocketController()
export class GameController {
  private getSocketGameRoom(socket: Socket): string {
    const socketRooms = Array.from(socket.rooms.values()).filter(
      (r) => r !== socket.id

    );
    const gameRoom = socketRooms && socketRooms[0];
    return gameRoom;
  }

  @OnMessage("update_game")
  public async updateGame(
    @SocketIO() io: Server,
    @ConnectedSocket() socket: Socket,
    @MessageBody() message: any
  ) {

    const gameRoom = this.getSocketGameRoom(socket);
    socket.to(gameRoom).emit("on_game_update", message);
  }
}
  @OnMessage("update_chat")
  public async updateChat(
    @SocketIO() io: Server,
    @ConnectedSocket() socket: Socket,
}
```

кінець лістингу 3.4

Файлова структура веб-платформи ілюструє логічну організацію проекту, підтримці та масштабуванні серверної частини системи. На основі рисунка 3.2 відображено оптимально розташовані компоненти та модулі, що дозволяють ефективно керувати функціональними можливостями та ресурсами платформи, забезпечуючи гнучкість та надійність в процесі експлуатації.

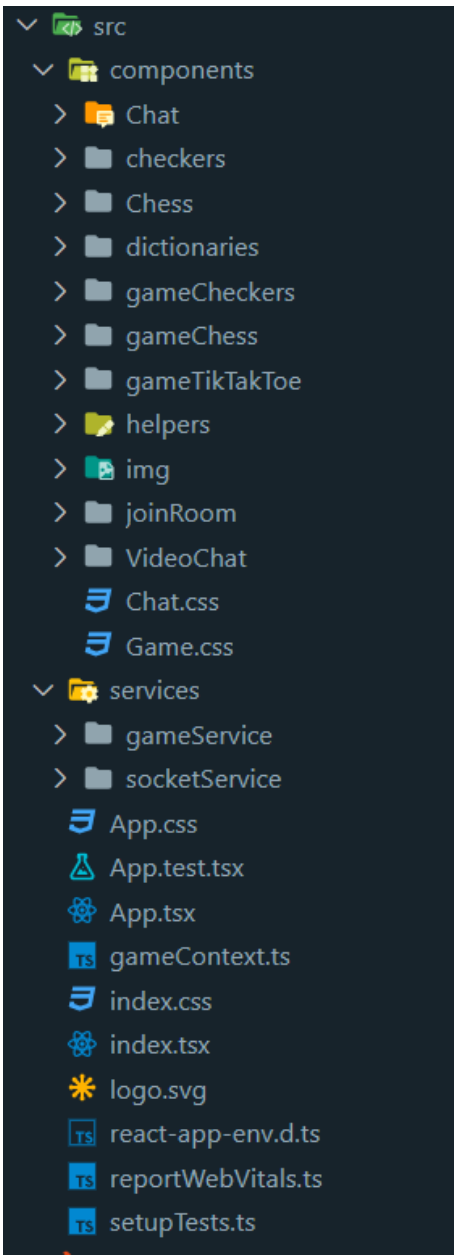

Рисунок 3.2 – Файлова структура проекту

Як і з сервером, запустимо середовище розробки Visual Studio Code. Створимо папки для розподілу компонентів для зручності використання.

У файлі «socketService > index.ts» створюємо підключення до локального серверу, зображено в лістингу 3.5.

Лістинг 3.5 – Система регенерації

```
class SocketService {
  public socket = io("http://localhost:9000/");
  public connect(
    url: string
```

```
  ): Promise<Socket<DefaultEventsMap, DefaultEventsMap>> {
    return new Promise((rs, rj) => {
      if (!this.socket) return rj();
      this.socket.on("connect", () => {
        rs(this.socket as Socket);
      });
      this.socket.on("connect_error", (err) => {
        console.log("Connection error: ", err);
        rj(err);
      });
    });
  }
}
export default new SocketService();
```

кінець лістингу 3.5

Файл «joinRoom -> index.tsx» один із ключових файлів. В ньому представлено ігри, які користувачі можуть вибрати, а також створення кімнати для гри. Для того щоб створити кімнату та розпочати гру, потрібно вибрати одну із представлених ігор та ввести ім’я кімнати у відповідному місці, після чого можна створити кімнату. Кімнату не буде створено, якщо не буде вибрана жодна із ігор або не буде написано ім’я кімнати.

Якщо один із користувачів створив кімнату, він повинен повідомити назву створеної ним кімнати і вибрану гру своєму знайомому/другу, який у свою чергу повинен обрати таку ж гру і ввести таку ж саму назву кімнати. Тільки після цього 2 користувачі приєднаються до кімнати і можуть розпочати гру. Якщо назва кімнати буде введена некоректно, або буде вибрана інакша гра, то створюється нова кімната і користувач буде чекати на підключення іншого користувача.

Приклад коду приєднання користувача до кімнати наведено в лістингу 3.6.

Лістинг 3.6 – Приєднання користувача до кімнати

```
const joinRoom = async (e: React.FormEvent) => {
  e.preventDefault();
  if (selectGame == "" || selectGame == null) return;
  const socket = socketService.socket;
  if (!roomName || roomName.trim() === "" || !socket) return;
  setJoining(true);
  const joined = await gameService
    .joinGameRoom(socket, roomName)
```

```
    .catch((err) => {
      alert(err);
    });
  if (joined) setInRoom(true);

  setJoining(false);
};
```

кінець лістингу 3.6

3.2 Тестування та налагодження ігрових механік

Після реалізації основної логіки клієнтської та серверної частини вебплатформи було проведено тестування ігрових механік з метою перевірки коректності взаємодії між гравцями та стабільності з’єднання в режимі реального часу. Особлива увага приділялася:

– правильності черговості ходів – система гарантує, що кожен гравець може здійснити хід лише під час своєї черги;

– валідації правил гри – сервер відхиляє некоректні ходи (наприклад, якщо гравець намагається перемістити фігуру супротивника або виконати хід, що суперечить шаховим правилам);

– синхронізації ігрової дошки – усі дії одного гравця миттєво відображаються на клієнті другого гравця завдяки використанню подій Socket.IO;

– реагуванню на завершення гри – система коректно визначає умови «шах», «мат» або «нічию», та надсилає обом гравцям подію «game_over» із відповідним повідомленням.

У процесі тестування використовувались як симуляції взаємодії з двох різних браузерів, так і ручні перевірки сценаріїв гри: початок партії, відмова від гри, передача ходу, обрив з’єднання тощо.

Також перевірялась робота інтерфейсу при високому навантаженні та втраті підключення: у випадку розриву з’єднання сервер зберігає стан гри, що дозволяє гравцю повторно підключитися без втрати прогресу.

Результати тестування підтвердили стабільну та коректну роботу основної логіки гри, що дозволяє розпочати процес розгортання платформи на хостинговому сервері для подальшого використання.

Далі на сервері виконується функція «joinGame». В даній функції перевіряється, чи до кімнати приєднані користувачі. Якщо при спробі приєднатися до кімнати в котрій вже присутні 2 гравці, з'явиться відповідне повідомлення, де буде сказано що кімнада за вказаною користувачем id переповнена. Якщо у кімнаті знаходиться один користувач, тоді в такому випадку можна приєднатись до кімнати і розпочинається гра. Дана функція наведена в лістингу 3.7.

Лістинг 3.7 – Функція для перевірки чи кімната переповнена/початок гри

```
public async joinGame(
  @SocketIO() io: Server,
  @ConnectedSocket() socket: Socket,
  @MessageBody() message: any
) {
  console.log("New User joining room: ", message);

  const connectedSockets =
io.sockets.adapter.rooms.get(message.roomId);
  const socketRooms = Array.from(socket.rooms.values()).filter(
    (r) => r !== socket.id
  );
  if (
    socketRooms.length > 0 ||
    (connectedSockets && connectedSockets.size === 2)
  ) {
    socket.emit("room_join_error", {
      error: "Room is full please choose another room to play!",
    });
  } else {
    await socket.join(message.roomId);
    socket.emit("room_joined");

    if (io.sockets.adapter.rooms.get(message.roomId).size === 2) {
      socket.emit("start_game", { start: true, symbol: "1" });
      socket
        .to(message.roomId)
        .emit("start_game", { start: false, symbol: "2" })
  }
```

кінець лістингу 3.7

3.3 Реалізація логіки взаємодії гравців у режимі онлайн

Однією з ключових функціональних частин вебплатформи є підтримка онлайн-гри між двома гравцями в режимі реального часу. Для цього було реалізовано логіку обміну даними через WebSocket-з'єднання з використанням бібліотеки Socket.IO [9]. Основна мета – забезпечити миттєву синхронізацію ходів між клієнтами, зберігаючи цілісність гри та правильність послідовності дій.

Під час створення або приєднання до шахової кімнати користувач передає дані на сервер, який виконує перевірку валідності імені кімнати, доступності та поточного статусу гри. Після успішного приєднання обом гравцям призначаються кольори фігур (чорні або білі), а їхні сеанси синхронізуються за допомогою унікального ідентифікатора кімнати.

Передача кожного ходу відбувається через подію «move_piece», що містить координати початкової та кінцевої позиції фігури. Сервер у свою чергу здійснює ретрансляцію цього ходу іншому гравцеві, гарантуючи одночасне оновлення стану дошки в обох клієнтів. На рисунку 3.3 зображено структуру обробки цієї події.

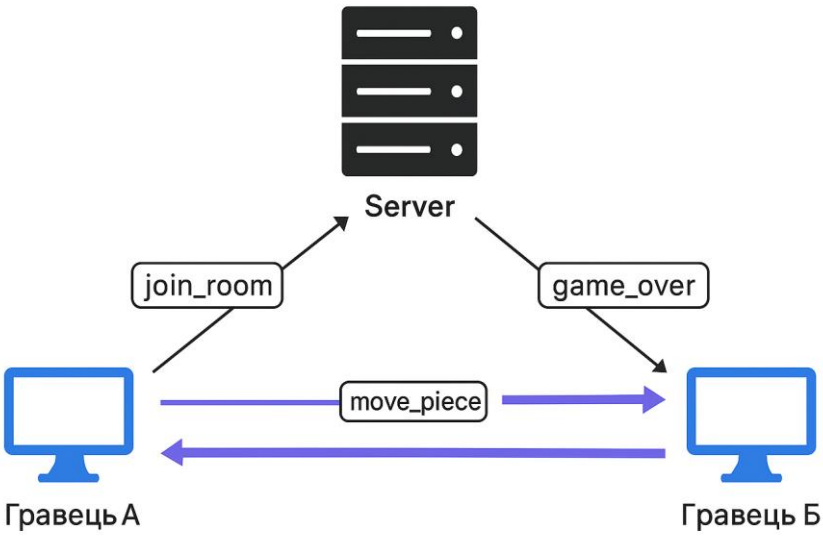

Рисунок 3.3 – Схема обміну ходами між гравцями через Socket.IO

У процесі розробки вебплатформи було передбачено обробку виняткових ситуацій, які можуть виникнути під час сеансу гри між користувачами. Зокрема, система реагує на випадки передчасного виходу гравця з кімнати, втрату з’єднання або спробу зробити хід у момент, коли черга належить іншому гравцеві. Для цього серверна логіка постійно контролює статус кожного підключення та своєчасно оновлює інформацію про стан гри. У разі таких подій система надсилає відповідні повідомлення другому гравцеві, інформуючи його про зміну ситуації – наприклад, про від’єднання опонента або завершення поточної сесії. Такий підхід дозволяє зберігати цілісність ігрового процесу, мінімізуючи помилки та непорозуміння між користувачами. У разі досягнення одного з фінальних результатів (оголошення мату або нічиєї), система автоматично надсилає подію «game_over» обом гравцям, супроводжуючи її повідомленням про результат гри.

На рисунку 3.4 представлено приклад активної гри між двома користувачами, з виведенням інформації про останній хід і поточний статус партії.

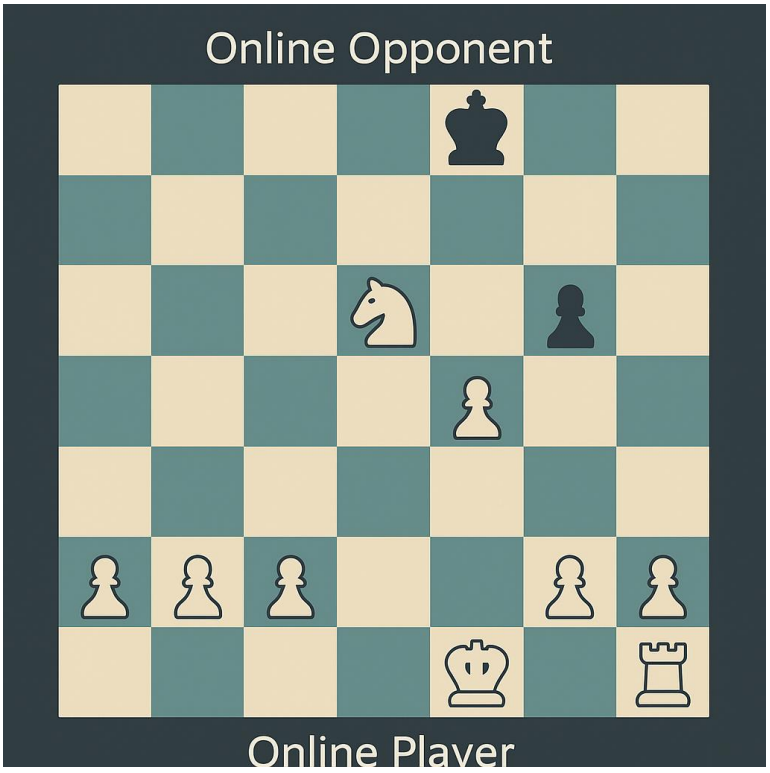

Рисунок 3.4 – Візуалізація онлайн-гри між двома гравцями

3.4 Створення графічного інтерфейсу та побудова ігрового рівня

Вибране програмне середовище Visual Studio Code для реалізації інтерфейсу користувача. Visual Studio Code – Інструмент для створення, редагування та налагодження сучасних веб-додатків і програм для хмарних систем. Visual Studio Code поширюється безкоштовно з версіями для Windows, Linux і OS X [10].

Продукт підтримує розробку на платформах ASP.NET і Node.js і позиціонується як легке рішення, яке не вимагає повністю інтегрованого середовища розробки. Підтримувані мови та технології включають: JavaScript, C++, C#, TypeScript, jade, PHP, Python, XML, Batch, F# DockerFile, Coffee Script, Java, HandleBars, R, Objective-C, PowerShell, Luna, Visual Basic, Markdown , JSON, HTML, CSS, LESS і SASS, Haxe [10].

Малюнки вказані нижче, будуть показувати архітектуру веб-платформи та взаємодію користувача з інтерфейсом.

Дана веб-платформа дозволить користувачам грати в представлені ігри один з одним та комфортно спілкуватися між собою під час гри. Інтерфейс веб-платформи буде простим і доступним, тобто у користувачів не виникне проблем із вивченням функціоналу даного сайту. На рисунку 3.5 показаний вигляд головного екрану гри.

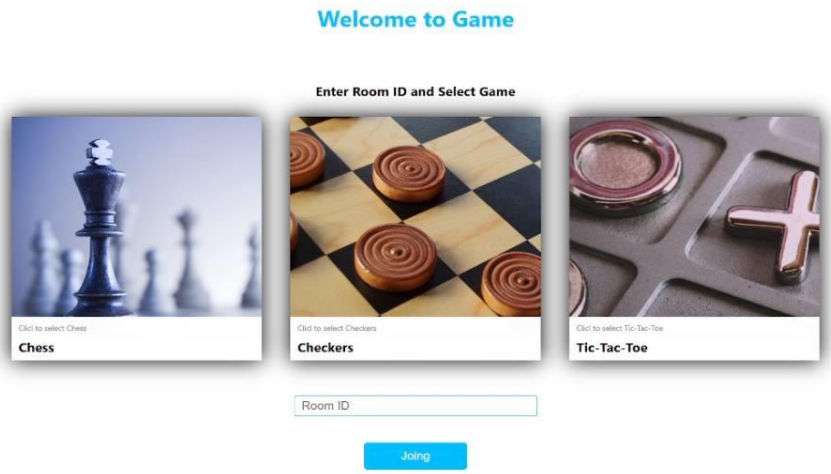

Рисунок 3.5 – Вигляд головного меню

Як вже було описано вище, на даній сторінці користувач вибирає одну із представлених ігор і створює кімнату для гри.

При наведені на одну із ігор, виїжджає невеликий опис, для загального розуміння правил гри, показано на рисунку 3.6.

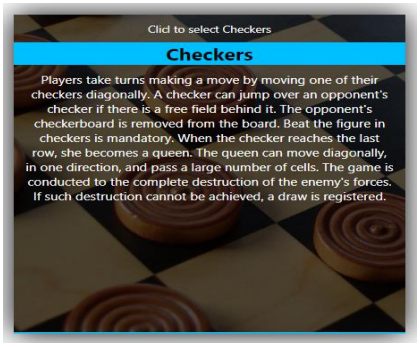

Рисунок 3.6 – Короткий опис гри при наведенні на неї

Після того як користувач увійшов у кімнату, йому потрібно дочекатися підключення опонента. Поки другий користувач не приєднався до кімнати, у першого на екрані буде відповідне повідомлення, що потрібно чекати підключення другого гравця. Вигляд даного повідомлення наведено на рисунку 3.7.

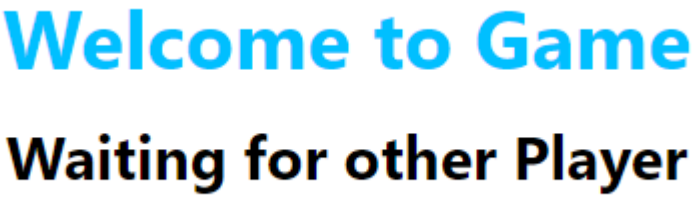

Рисунок 3.7 – Повідомлення про очікування підключення опонента

Приєднання гравців один до одного у кімнаті, як і самі ігри реалізовані з допомогою Socket.io.

Socket.IO – бібліотека JavaScript для веб-додатків і обміну даними в режимі реального часу. Він складається з двох частин: клієнта, що працює в браузері, і

сервера в node.js. Обидва компоненти мають схожі інтерфейси прикладного програмування. Як і node.js, Socket.IO орієнтований на події.

Socket.IO в основному використовує протокол Web-Socket, але за бажанням можна використовувати інші методи, такі як сокети Adobe Flash, запити JSON або запити AJAX, щоб забезпечити той самий інтерфейс [11]. На додаток до того факту, що Socket.IO можна використовувати як оболонку для Web-Sockets, він містить багато інших функцій, включаючи підвішування на кількох сокетах, зберігання даних, пов’язаних з кожним клієнтом, та асинхронний ввід-вивід [12].

За допомогою socket.io створюються кімнати, показано на рисунку 3.8.

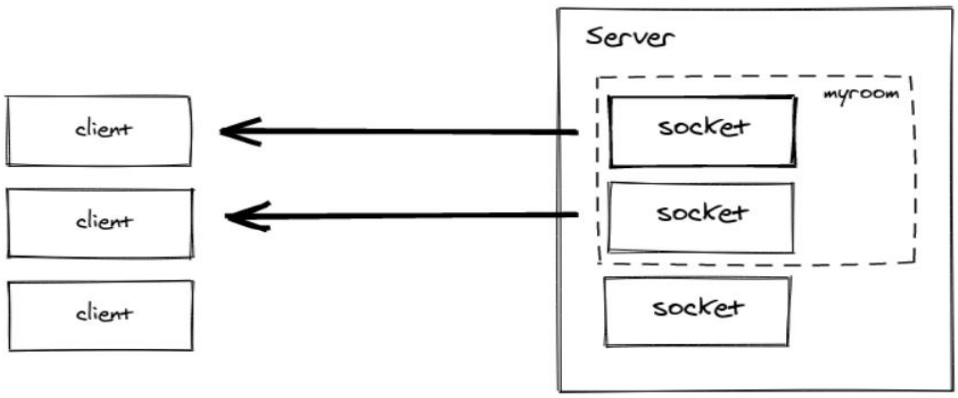

Рисунок 3.8 – Ілюстрація створення кімнати [13]

Користувач, вписавши назву (id) кімнати, активує функцію під назвою «joinGameRoom», яка приймає в себе сокет користувача та вписане ним id кімнати (ліст. 3.8).

Лістинг 3.8 – Функція приєднання до кімнати

```
public async joinGameRoom(socket: Socket, roomId: string):
  return new Promise((rs, rj) => {
    socket.emit("join_game", { roomId });
    socket.on("room_joined", () => rs(true));
    socket.on("room_join_error", ({ error }) => rj(error));
  });
};
```

кінець лістингу 3.8

3.5 Розробка відео-чату

Відео-чат створений також з допомогою socket.io. На сервері створена функції для дзвінку, а саме кнопка, відповідь на дзвінок, закінчити дзвінок, показано в лістингу 3.9 [14].

Лістинг 3.9 – Реалізація дзвінку, відповідь, закінчити дзвінок

```
io.on("connection", (socket) => {
  socket.emit("me", socket.id);
  socket.on("disconnect", () => {
    socket.broadcast.emit("callEnded");
  });
  socket.on("callUser", (data) => {
    io.to(data.userToCall).emit("callUser", {
      signal: data.signalData,
      from: data.from,
      name: data.name,
    });
  });
  socket.on("answerCall", (data) => {
    io.to(data.to).emit("callAccepted", data.signal);
  });
;};
```

кінець лістингу 3.9

Для початку відеозв’язку гравцям необхідно натиснути кнопку «Start VideoChat», після чого на екрані з’являється попередній перегляд з камери користувача, а також кнопка для здійснення дзвінка опоненту. У разі натискання цієї кнопки, на стороні опонента автоматично з’являється повідомлення про вхідний виклик із пропозицією прийняти дзвінок. При підтвердженні з’єднання опонентом, обидва учасники гри отримують доступ до відеозв’язку – зображення з камери співрозмовника транслюється у реальному часі, а також активується передача голосу. Таким чином забезпечується двостороння комунікація між гравцями, що сприяє більш реалістичному ігровому досвіду. Фрагмент реалізації цього функціоналу наведено у лістингу 3.10 [15].

Лістинг 3.10 – Створення камери

```
if (startVideoChat == true) {
  navigator.mediaDevices.getUserMedia({ video: true, audio: true
}).then((stream: any) => {
    setStream(stream);
    myVideo.current.srcObject = stream;
  });
}
showSocketId();
if (socketService.socket?.id) {
  setMe(socketService.socket?.id);
};
```

кінець лістингу 3.10

На рисунку 3.9 зображено інтерфейс сповіщення користувача про вхідний дзвінок від опонента у режимі відеозв'язку. Під час ініціації дзвінка з боку іншого гравця на екрані з'являється вікно з повідомленням «Oponent is calling...» та кнопкою «Answer», яка дозволяє прийняти виклик. Інтерфейс слугує зручним інструментом для налагодження комунікації під час гри.

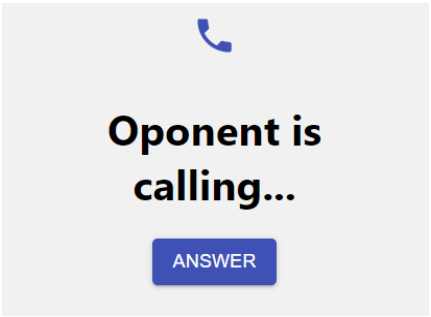

Рисунок 3.9 – Дзвінок від опонента

Висновки до розділу 3

У третьому розділі було реалізовано повноцінну веб-платформу для багатокористувацької гри в шахи з використанням бібліотеки React для клієнтської частини та Socket.IO для забезпечення реального часу в обміні даними між користувачами. Було розроблено основну ігрову логіку, реалізовано механізм створення та приєднання до ігрових кімнат, обробку ходів гравців,

перевірку правил шахової гри та фіксацію завершення партії. Також розроблено систему обміну повідомленнями та передачі ігрового стану між користувачами через веб-сокети.

Важливим етапом стала розробка користувацького інтерфейсу гри, який забезпечує зручне відображення дошки, фігур, історії ходів та статусу гравців. Окрему увагу було приділено створенню адаптивного дизайну для різних пристроїв.

Було проведено три типи тестування: юніт-тестування окремих функцій гри, інтеграційне тестування взаємодії компонентів та тестування продуктивності гри за умов одночасного підключення кількох користувачів. Отримані результати засвідчили стабільну роботу платформи та відповідність функціональності поставленим вимогам.

Фінальним етапом стала підготовка та збірка проєкту для розгортання на сервері, що включало створення інсталяційного пакету та опис процесу його налаштування.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи бакалавра були досягнуті такі основні результати:

1) проведено аналіз сучасних онлайн-рішень для багатокористувацьких шахових ігор, що дало змогу виявити основні тенденції в області застосування вебтехнологій, а також визначити сильні сторони й недоліки існуючих платформ за критеріями технологічної реалізації, дизайну та зручності використання;

2) досліджено можливості фреймворку React для розробки динамічного користувацького інтерфейсу із сучасними інтерактивними елементами, зокрема реалізовано функціонал вибору гри, введення Room ID і з'єднання з іншими гравцями, що сприяє високій гнучкості та адаптивності платформи;

3) розроблено серверну частину платформи на базі Node.js із використанням бібліотеки Socket.IO, що забезпечило обмін подіями між клієнтами в режимі реального часу, організацію ігрових кімнат, фіксацію ходів та контроль завершення ігор;

4) створено сучасний, адаптивний та привабливий інтерфейс користувача відповідно до стандартів UI/UX-дизайну, що забезпечує інтуїтивно зрозумілу навігацію і простоту входу в гру для різних категорій користувачів;

5) реалізовано основні ігрові механіки для класичних настільних ігор – шахів і шашок, зокрема впроваджено логіку пересування фігур, чергування ходів і перевірку коректності ігрового процесу;

6) забезпечено підключення до гри через унікальний Room ID, що дозволяє організовувати як приватні, так і публічні ігрові сесії, підвищуючи варіативність і доступність платформи;

7) проведено тестування розробленої платформи, що включало перевірку стабільності з'єднання, коректності ігрової логіки та адаптивності інтерфейсу на різних пристроях, у результаті чого підтверджено відповідність продукту заявленим вимогам.

У підсумку, в межах виконаної роботи створено функціональний прототип вебплатформи для багатокористувацької гри в шахи з можливістю подальшого розширення функціоналу, що забезпечує зручну, швидку й стабільну взаємодію між гравцями у реальному часі на основі сучасних вебтехнологій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Security aspects of full-duplex web interactions and WebSockets. 2023 20th ACS/IEEE International Conference on Computer Systems and Applications. URL: https://doi.org/10.1109/aiccsa59173.2023.10479302 (дата звернення: 15.03.2025).

2. Lichess.org – free open source chess platform. URL: https://lichess.org (дата звернення: 31.05.2025).

3. Fitriani L., Tresnawati D., Pamungkas M. I. I. S. Multiplayer Game Guessing Sundas Proverb Using Socket.Io And Node.Js. JUITA: Jurnal Informatika. 2023. Т. 11, № 2. С. 231. URL: https://doi.org/10.30595/juita.v11i2.16828 (дата звернення: 27.05.2025).

4. React – official documentation. URL: https://reactjs.org (дата звернення: 03.05.2025).

5. Socket.IO – Real-time bidirectional event-based communication. URL: https://socket.io/docs/v4/ (дата звернення: 03.06.2025).

6. Bin Uzayr S. Vue Components. Mastering Vue.js. Boca Raton, 2022. С. 59-96. URL: https://doi.org/10.1201/9781003310464-3 (дата звернення: 03.06.2025).

7. Visual Studio Code – Code Editing. Redefined. Microsoft. URL: https://code.visualstudio.com/docs (дата звернення: 03.06.2025).

8. Socket.IO – official documentation. URL: https://socket.io/docs/ (дата звернення: 31.05.2025).

9. Socket.IO – Rooms and namespaces. Official documentation. URL: https://socket.io/docs/v4/rooms/ (дата звернення: 03.05.2025).

10. Krasnovidov A. V., Khomonenko A. D. Integration of MatLab and R with high-level languages using C# and Microsoft Visual Studio as an example. Journal of Physics. Conference Series. 2021. Т. 2131 № 2. С. 022096. URL: https://doi.org/10.1088/1742-6596/2131/2/022096 (дата звернення: 15.05.2025).

11. The Impact of Limited Prosthetic Socket Documentation. A Researcher Perspective / J. Olsen and ek. Frontiers in Rehabilitation Sciences. 2022. T. 3. URL: https://doi.org/10.3389/fresc.2022.853414 (дата звернення: 04.05.2025).

12. Cantelon M., Harter M., Holowaychuk T., Rajlich N. Node.js in Action. Manning Publications. Real-time web with Socket.IO (дата звернення: 14.05.2025).

13. Socket.IO – Rooms and namespaces. URL: https://socket.io/docs/v4/rooms/ (дата звернення: 01.05.2025).

14. Topalian C. JavaScript Bookmarklet Advanced Programming. Unleash the Power of JS by Learning How to Modify Any Website and Learn How to Make Your Own Custom YouTube Video Interfaces. (дата звернення: 11.05.2025).

15. GeeksforGeeks: in the article – how to make a video call app in Node.js. URL: https://www.geeksforgeeks.org/how-to-make-a-video-call-app-in-node-js/ (дата звернення: 15.05.2025).