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 веб-сайту для онлайн гри «Мафія» з використанням фреймфорку React та Node.js

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

(підпис) Корлиханов О.С. (прізвище та ініціали)

Керівник (підпис) Мудрик І.Я. (прізвище та ініціали)

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

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

Рецензент (підпис) Марценко С. В. (прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра Кафедра програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Петрик М.Р.
(підпис) (прізвище та ініціали)
« » 20___ р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

студенту Корлиханову Олександру Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка веб-сайту для онлайн гри «Мафія» з фреймворку React та Node.js

Керівник роботи Мудрик Іван Ярославович, доктор філософій
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « » 20___ року №

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи

4. Зміст роботи (перелік питань, які потрібно розробити)
Аналіз предметної області розробки веб-додатків, Огляд програм-аналогів, Обґрунтування вибору напряму дослідження, Методологія розробки, Формування вимог до системи, Проєктування веб-застосунку, Конструювання веб-застосунку, Робота із Git,Тестування та верифікація вимог, Соціальні та психологічні фактори ризику в умовах IT-середовища, Заходи щодо захисту обладнання від короткого замикання

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
діаграма варіантів використання, діаграма архітектури системи, діаграма сутностей, діаграма послідовності головного сценарію гри, діаграма модулів

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка

Студент _____ *Корлиханов О.С.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Мудрик І.Я.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Корлиханов Олександр Сергійович Розробка веб-сайту для онлайн гри «Мафія» з фреймворку React та Node.js. Кваліфікаційна робота – Тернопільський національний технічний університет імені Івана Пулюя. Факультет комп'ютерно-інформаційних систем та програмної інженерії, група СП-41. Тернопіль: ТНТУ, 2025 р., сторінок – 81, джерел – 33, рисунків – 23, таблиць – 11, додатків – 4.

Ключові слова: веб-застосунок, онлайн-гра, React, Node.js, WebSocket, багатокористувацька система, Мафія.

Метою кваліфікаційної роботи є створення функціонального веб-сайту для гри «Мафія» з підтримкою взаємодії в реальному часі, автоматизацією ігрових процесів, зручним інтерфейсом користувача та можливістю гнучкого налаштування параметрів гри. Основним завданням є розробка SPA-застосунку, що забезпечує розподіл ролей, хід гри, нічні дії, голосування та визначення переможців без участі ведучого. Для досягнення мети використано методи компонентної архітектури, технології WebSocket та REST API, а також засоби фронтенду (React) та бекенду (Node.js, NestJS). Система розроблена з урахуванням сучасних вимог до інтерактивності, масштабованості та зручності використання.

У роботі проаналізовано існуючі аналоги, спроєктовано архітектуру, реалізовано і протестовано застосунок. Увагу приділено технічним та експлуатаційним характеристикам, зокрема швидкодії та підтримці багатьох користувачів. Результати роботи можуть бути впроваджені як у приватному порядку, так і на публічних платформах. Проєкт має соціальну значимість як інструмент для популяризації українського контенту та розвитку онлайн-комунікації.

У підсумку, робота демонструє можливість ефективного застосування сучасних веб-технологій для створення багатокористувацьких інтерактивних систем і може бути основою для подальшого розвитку подібних продуктів.

ABSTRACT

Korlykhanov Oleksandr Serhiyovych Development of a website for the online game "Mafia" using Node.js technologies and the framework - React. Qualification work - Ivan Pulyuy Ternopil National Technical University. Faculty of Computer Information Systems and Software Engineering, group SP-41. Ternopil: TNTU, 2025, pages - 81, sources - 33, figures - 23, tables - 11, appendices - 4.

Keywords: web application, online game, React, Node.js, WebSocket, multi-user system, Mafia.

The purpose of the qualification work is to create a functional website for the game "Mafia" with support for real-time interaction, automation of game processes, a convenient user interface and the ability to flexibly configure game parameters. The main task is to develop an SPA application that provides role distribution, game progress, night actions, voting and determining winners without the participation of the host. To achieve the goal, component architecture methods, WebSocket and REST API technologies, as well as frontend (React) and backend (Node.js, NestJS) tools were used. The system was developed taking into account modern requirements for interactivity, scalability and ease of use.

The work analyzed existing analogues, designed the architecture, implemented and tested the application. Attention was paid to technical and operational characteristics, in particular, speed and support for many users. The results of the work can be implemented both privately and on public platforms. The project has social significance as a tool for popularizing Ukrainian content and developing online communication.

In conclusion, the work demonstrates the possibility of effective application of modern web technologies to create multi-user interactive systems and can be the basis for further development of similar products.

ЗМІСТ

ВСТУП 7

РОЗДІЛ 1. ОГЛЯД І АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗРОБКИ ВЕБ-ДОДАТКІВ ДЛЯ БАГАТОКОРИСТУВАЦЬКИХ ОНЛАЙН ІГОР 9

1.1 Аналіз предметної області розробки веб-додатків 9

1.2 Огляд програм-аналогів 10

1.3 Обґрунтування вибору напряму дослідження 13

1.4 Методологія розробки 14

1.5 Формування вимог до системи 15

РОЗДІЛ 2. РОЗРОБКА МОДЕЛІ ТА ПРОГРАМНОГО КОМПЛЕКСУ 21

2.1 Проєктування веб-застосунку 21

2.2 Конструювання веб-застосунку 35

2.3 Робота із Git 53

2.4 Тестування та верифікація вимог 55

2.5 Результати розробки 59

РОЗДІЛ 3. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ТА ОХОРОНИ ПРАЦІ 60

3.1 Соціальні та психологічні фактори ризику в умовах ІТ-середовища 60

3.2 Заходи щодо захисту обладнання від короткого замикання 61

ВИСНОВКИ 63

ВИКОРИСТАННІ ДЖЕРЕЛА 65

ДОДАТОК А 70

ДОДАТОК Б 75

ДОДАТОК В 79

ДОДАТОК Г 81

ВСТУП

У сучасному світі цифрових технологій спостерігається постійне зростання попиту на інтерактивні веб-застосунки, зокрема в сфері онлайн-розваг. Одним із популярних жанрів серед користувачів різного віку є соціально-психологічні ігри, такі як «Мафія». Ця гра поєднує елементи дедукції, психології та командної взаємодії, що робить її захопливою як у традиційному, так і в онлайн-форматі. Проте, незважаючи на популярність гри серед користувачів різного віку та з різних країн, більшість існуючих веб-платформ для гри в «Мафію» мають певні обмеження. Часто вони орієнтовані виключно на англомовну аудиторію, що створює труднощі для користувачів з інших мовних середовищ. Крім того, деякі з таких платформ мають обмежену функціональність, заплутаний або застарілий інтерфейс, недостатню зручність користування або не передбачають автоматизацію ігрового процесу.і відсутність підтримки української мови. Це створює актуальний запит на створення україномовного, зручного та технічно досконалого веб-застосунку для проведення гри «Мафія» онлайн.

Розвиток технологій веб-програмування, зокрема JavaScript-екосистеми, відкриває нові можливості для розробки динамічних веб-додатків із підтримкою багатокористувацької взаємодії в реальному часі. Використання фреймворку React для клієнтської частини дозволяє створити швидкий і адаптивний інтерфейс, тоді як Node.js у поєднанні з NestJS на стороні сервера забезпечує ефективну обробку запитів та підтримку WebSocket-з’єднань. Завдяки цьому можна реалізувати логіку гри, де кожен гравець отримує роль, взаємодіє з іншими, бере участь у голосуваннях та нічних діях – усе це в режимі реального часу.

У межах даної кваліфікаційної роботи об’єктом дослідження є процес створення веб-застосунку, призначеного для організації багатокористувацької гри онлайн. Особлива увага приділяється моделюванню ігрового процесу, керуванню діями користувачів, розподілу ролей, реалізації фаз гри, а також підтримці

голосувань і взаємодії між гравцями. Предметом дослідження є архітектурні, програмні та інтерфейсні рішення, які дозволяють відтворити гру «Мафія» в інтерактивному веб-середовищі з максимальним наближенням до її класичної версії, водночас використовуючи переваги автоматизації.

Предметом дослідження є архітектурні, програмні та інтерфейсні рішення, що дозволяють реалізувати гру «Мафія» в інтерактивному веб-середовищі.

Основною метою цієї роботи є розробка повноцінного веб-додатку для гри в «Мафію» з можливістю онлайн-участі без залучення ведучого. Передбачається, що додаток дозволятиме автоматично розподіляти ролі між гравцями, забезпечувати підтримку основних і спеціальних фаз гри, фіксувати статистику гравців, надавати інтерфейс для голосувань, відображати події гри та взаємодію в реальному часі. Важливою особливістю є також локалізація інтерфейсу українською мовою та забезпечення інтуїтивно зрозумілого, дружнього до користувача дизайну.

Для досягнення поставленої мети в роботі вирішуються наступні завдання:

- провести аналіз предметної області та існуючих аналогів гри «Мафія»;
- визначити вимоги до функціональності та інтерфейсу системи;
- спроєктувати архітектуру клієнтської і серверної частин;
- реалізувати ключову логіку гри (фази, дії ролей, голосування);
- забезпечити збереження та обробку статистичних даних гравців;
- протестувати систему на відповідність вимогам.

Розробка такого застосунку є актуальною з кількох причин. По-перше, вона відповідає тенденціям розвитку веб-технологій та індустрії онлайн-ігор та по-друге, вона покриває потребу україномовної аудиторії в локалізованому й доступному рішенні. Таким чином, проєкт поєднує в собі як інноваційний технічний підхід, так і соціально значущу складову.

Структура кваліфікаційної роботи включає: вступ; теоретичний розділ із аналізом предметної області та технологій; практичний розділ із проєктуванням та реалізацією застосунку; висновки та список використаних джерел.

РОЗДІЛ 1. ОГЛЯД І АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗРОБКИ ВЕБ-ДОДАТКІВ ДЛЯ БАГАТОКОРИСТУВАЦЬКИХ ОНЛАЙН ІГОР

1.1 Аналіз предметної області розробки веб-додатків

Розробка веб-сайтів і веб-додатків є однією з найактивніших галузей у сучасній IT-індустрії. Очікується, що у 2025 році обсяг цього ринку сягне приблизно 75 мільярдів доларів США. В межах цієї сфери існує значна кількість напрямів, серед яких особливо виділяються бізнес-сайти, особисті сторінки, інтернет-магазини, розважальні ресурси, освітні платформи, соціальні мережі та ігрові веб-додатки [1].

Найбільшу частку на ринку займають корпоративні сайти, що використовуються компаніями для представлення послуг, продуктів та зворотного зв’язку з клієнтами. Значною популярністю користуються й особисті веб-ресурси – блоги, онлайн-резюме та портфоліо. Електронна комерція представлена такими гігантами, як Amazon і Rozetka, і також формує вагому частину ринку.

Ігрові веб-додатки, хоча й мають відносно невелику частку, продовжують стабільно розвиватися. Цей сегмент став особливо актуальним у період пандемії COVID-19, коли зросла потреба у цифрових розвагах. Незважаючи на менший обсяг, розробка ігрових веб-додатків вимагає реалізації складних технологічних рішень: обробки даних у реальному часі, підтримки багатокористувацької взаємодії та розробки адаптивної логіки гри.

Таким чином, хоча ігрові веб-проєкти не є домінуючим напрямом у сфері веб-розробки, вони мають значний потенціал та інженерну цінність, особливо для фахівців, зацікавлених у вирішенні нетривіальних технічних задач.

1.2 Огляд програм-аналогів

З метою кращого розуміння ринку та визначення основних функціональних і технічних особливостей програмного продукту, було проведено огляд існуючих рішень, подібних за функціональністю до розроблюваного веб-сайту для онлайн гри «Мафія». В наслідок аналізу були знайдені проєкти-аналоги, які реалізують подібну ігрову механіку, підтримують багатокористувацьку взаємодію в режимі реального часу та забезпечують користувацький інтерфейс через веб-технології.

1) Аналіз проєкту “Mafia: The Game” [2]:

Загальна інформація:

- Назва: “Mafia: The Game”
- Розробник: Ben Robinson
- Платформи: Веб, iOS, iPadOS, macOS, tvOS
- Доступність:Безкоштовно, без реклами та внутрішніх покупок

Ключові функції:

- Можливість створення приватних ігор та приєднання за кодом
- Ролі гравці: мафія, детектив, лікар, шериф, мирні жителі
- Голосування для виявлення мафії
- Режим "Storyteller" для додаткової розваги

Технічні аспекти:

- Фронтенд: React.js
- Бекенд: Node.js з Express.js
- Хостинг: Netlify (фронтенд), Heroku (бекенд)
- Мобільні додатки: SwiftUI для iOS, iPadOS, macOS, tvOS

Переваги:

- Кросплатформеність
- Адаптивність
- Простота у використанні

- Режим "Storyteller" додає інтерактивності

Недоліки:

- Відсутність вбудованого чату, залежність від зовнішніх засобів зв’язку
- Відсутність системи облікових записів та збереження прогресу
- Обмеженість вибору додаткових ролей
- Примітивний UI

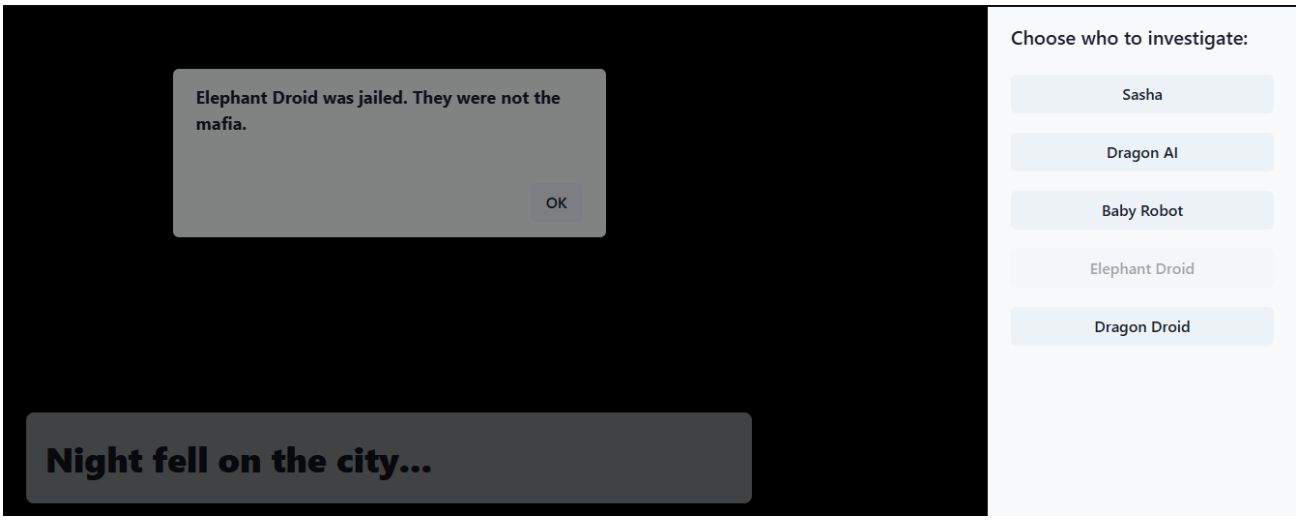

Рисунок 1.1 – Вигляд інтерфейсу

2) Аналіз проєкту “Mafia.gg” [3]:

Загальна інформація:

- Назва: “Mafia.gg”
- Розробник: Спільнота пов’язана іщ iateyourpie
- Платформа: Веб
- Доступність: Необхідність реєстрації

Ключові функції:

- Текстовий геймплей
- Різноманіття ролей
- Гнучке налаштування ігрової сесії

Технічні аспекти:

- Точна інформація невідома, проте можна припустити використання React або Vue для фронтенд розробки та Node.js або Python для бекенду.

Переваги:

- Гнучкість
- Активна спільнота
- Велика варіаційність

Недоліки:

- Високий поріг входу для новачків
- Складний UI

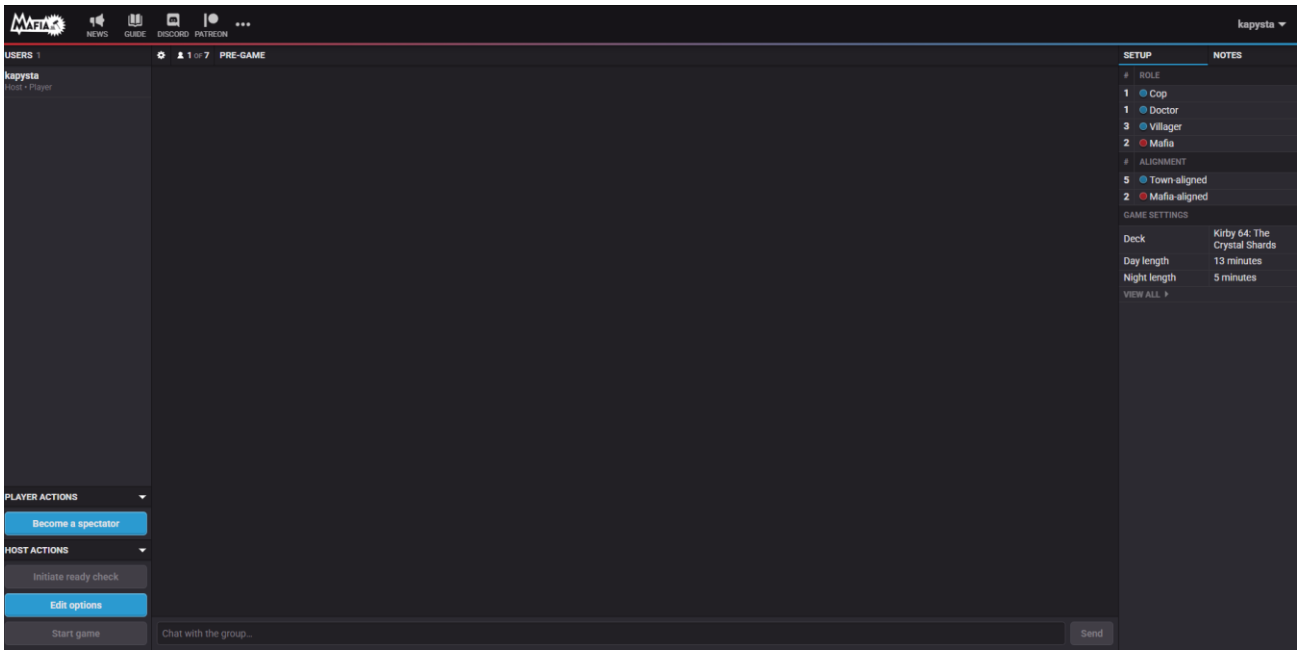

Рисунок 1.2 – Вигляд інтерфейсу

Провівши огляд та аналіз програм-аналогів, було визначено їх ключові функціональні особливості, переваги та недоліки. Це дозволило сформувати вимоги до власного веб-додатку, враховуючи найкращі практики реалізації гри «Мафія», а також уникнути типових недоліків існуючих рішень. Зокрема, було акцентовано увагу на зручності інтерфейсу, доступності гри без реєстрації а також можливості створення приватних ігор з гнучкими налаштуваннями.

1.3 Обґрунтування вибору напряму дослідження

Зважаючи на стрімкий розвиток сфери онлайн-ігор та зростаючий попит на інтерактивні веб-застосунки, розробка платформи для гри «Мафія» є обґрунтованим і актуальним напрямом. Як зазначається у дослідженні (Корлиханов, Мудрик, 2025), незважаючи на велику кількість англомовних аналогів, україномовний ринок подібних продуктів залишається недостатньо насиченим, що створює додаткові можливості для розвитку локального IT-продукту.

Застосування сучасних технологій, зокрема Node.js та React, дозволяє ефективно реалізувати необхідну функціональність гри. Такі інструменти, як NestJS для побудови модульної серверної архітектури та WebSocket для забезпечення реального часу, є оптимальними для мультикористувацьких ігор. Вони дозволяють організувати чітке розмежування логіки, забезпечити стабільний обмін даними та масштабованість рішення [5].

Додатковою перевагою є використання PostgreSQL як основи для реляційної бази даних, що дозволяє ефективно управляти структурою сутностей гри, зокрема гравцями, кімнатами, ігровими діями тощо.

Також, як зазначено в наукових джерелах, важливим аспектом є побудова інтерфейсу на базі компонентної архітектури. Реалізація за допомогою React дозволяє не лише покращити продуктивність оновлень, а й спростити підтримку та масштабування проекту в майбутньому.

Таким чином, вибраний напрям дослідження — розробка україномовного веб-застосунку для гри «Мафія» з використанням сучасного стеку технологій — є обґрунтованим з огляду на як ринкові потреби, так і технічну доцільність реалізації.

1.4 Методологія розробки

У процесі створення програмного забезпечення важливо не лише визначити функціональні вимоги до продукту, а й обрати ефективну модель організації розробки. Методології розробки програмного забезпечення — це сукупність підходів, принципів і практик, що регламентують послідовність виконання етапів створення програмного продукту, а також забезпечують якість, надійність і ефективність кінцевого результату.

Кожна методологія має власні особливості, переваги та недоліки, які варто враховувати при виборі підходу до розробки. Правильний вибір методології дозволяє оптимізувати процес розробки, зменшити ризики, а також адаптувати проєкт до змін у вимогах або ринковій ситуації. Далі буде наведено список із методологій для зручного аналізу.

Таблиця 1.1 – Методології розробки

Методологія	Головна перевага	Головний недолік
Waterfall	Чітка структура та послідовність етапів	Неможливо повернутись до попередніх етапів без значних переробок
Spiral	Орієнтована на зменшення ризиків	Висока складність планування та реалізації
Agile	Висока гнучкість та адаптивність	Може призвести до хаосу при недостатній організації
Scrum	Чітка структура роботи в спринтах, дозволяє швидко помічати та виправляти проблеми	Потребує чіткого розподілу ролей, при маленькій команді не є виправданою
Kanban	Простота впровадження та наочна візуалізація задач	Відсутність чітких дедлайнів може призвести до затягування строків

Опираючись на наведені дані, можна зробити висновок, що методології Waterfall та Spiral не відповідали вимогам проєкту через свою негнучкість та

складність. Методологія Scrum також була недоцільною, оскільки передбачає наявність кількох чітко визначених ролей, що є зайвим при індивідуальній розробці. У процесі роботи було обрано комбінований підхід, що поєднує гнучкість Kanban із певними організаційними принципами Scrum. Такий гібрид дозволив ефективно управляти великою кількістю задач, але при цьому не потребував чітких командних ролей [6].

1.5 Формування вимог до системи

1.5.1 Загальні характеристики системи

Розроблювана система є веб-застосунком, призначеним для проведення онлайн-ігри «Мафія» в реальному часі. Основною метою проєкту є створення інтерактивного середовища, у якому користувачі зможуть брати участь у грі, взаємодіяти одне з одним, отримувати ролі та виконувати ігрові дії в межах сесії.

Система дозволяє автоматизовано організовувати гру «Мафія» забезпечуючи обробку нічних дій, голосувань, переходу між фазами та взаємодію гравців через вбудований чат. Користувачі можуть створювати або приєднуватись до ігрових лобі, проходити авторизацію та зручно керувати процесом гри.

При формуванні вимог було враховано дослідження описане у [7].

Основні цілі системи:

- Надати українськомовним користувачам доступ до гри «Мафія» в зручному онлайн-форматі.
- Автоматизувати правила гри: розподіл ролей, черговість дій, обробку результатів голосувань.
- Забезпечити можливість гри у реальному часі з іншими учасниками.
- Зберігати історію ігор та базову статистику.
- Підтримувати інтуїтивно зрозумілий інтерфейс із сучасним дизайном.

Основні ролі:

- Гравець – учасник гри, що може створити або приєднатись до лобі, брати участь у голосуванні та ігрових діях.
- Ведучий гри – користувач, що створює гру та має додаткові повноваження щодо налаштування параметрів
- Система (сервер) – автоматичний «ведучий», що контролює логіку гри, черговість фаз, нічні дії та результати голосування.

1.5.2 Функціональні вимоги

1) Функції авторизації та реєстрації користувача в системі гри
   - Створення облікового запису з нікнеймом та паролем.
   - Вхід до системи з подальшим збереженням сесії.
   - Можливість приєднання до гри без реєстрації як гість.
2) Створення та приєднання до лобі передбачає наступні функції:
   - Створення нового ігрового лобі з унікальним ідентифікатором.
   - Приєднання до наявного лобі за кодом.
   - Можливість обмежити доступ паролем або кількістю гравців.
   - Наявність списку активних ігор до який можна приєднатись.
3) Керування ігровим процесом (для адміністратора лобі) вимагає наявності наступних функцій:
   - Встановлення кількості гравців та ролей у грі.
   - Таймер для обмеження часу на хід, обговорення чи голосування.
   - Запуск гри після приєднання достатньої кількості учасників.
   - Можливість обирати між ШІ ведучим та ручним.
   - Можливість задати ролі гравцям ручним та автоматичним методом.
   - Можливість передати роль адміністратора гри іншому гравцю.
4) Геймплейна логіка має наступні вимоги:

- Автоматична зміна станів лобі в ігровому процесі
- Фаза ночі: можливість виконання ролями своїх дій (наприклад, мафія обирає жертву, лікар – кого лікувати).
- Фаза дня: обговорення, голосування за підозрюваних, оголошення результатів.
- Визначення переможців за правилами гри.

5) Внутрнішньоігровий чат повинен:
   - Бути загальним чатом у лобі.
   - Ставати обмеженим під час гри (в залежності від фази та ролі).
   - Слугувати інструментом для системних повідомлень про хід гри (ніч почалась, голосування завершено тощо).
6) Збереження та перегляд статистики передбачає:
   - Збереження базової статистики користувача (кількість перемог, зіграних ігор).
   - Можливість перегляду історії попередніх ігор

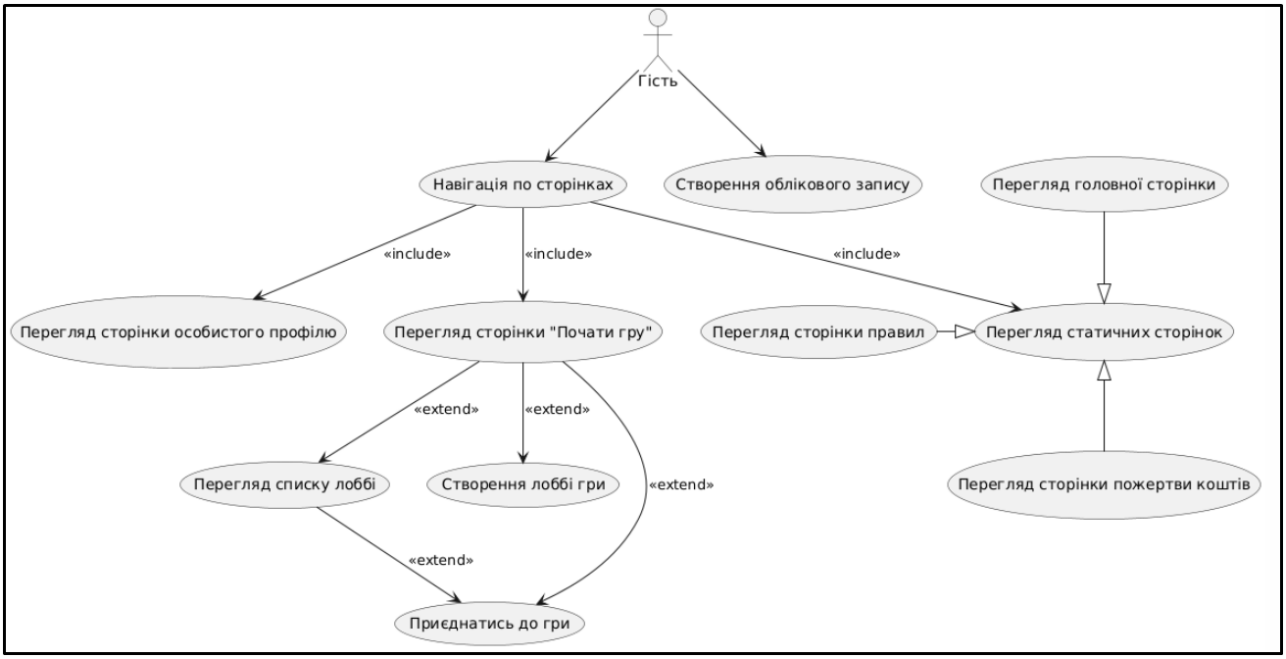

Рисунок 1.3 – Діаграма варіантів використання для актора Гість

На діаграмі зображеній на рисунку 1.3 показані функціональні можливості користувача в момент коли він зайшов на сайт. Варіанти використання: Перегляд сторінки правил, Перегляд сторінки пожертви коштів та Перегляд головної сторінки було зведено до одного Перегляд статичних сторінок для спрощення діаграми. Перегляд облікового запису передбачає попереднього його створення. Після виконання ВВ: Приєднатись до гри та Створення лоббі гри, стан користувача змінюється на Гравець та Ведучий відповідно.

1.5.3 Нефункціональні вимоги:

1) Продуктивність та масштабованість
   - Система повинна забезпечувати стабільну роботу при навантаженні до 20 одночасних гравців у грі.
   - Час реакції на основні дії (надсилання повідомлення, голосування, вибір дій у фазі ночі) не повинен перевищувати 300 мс при стандартному навантаженні.
2) Надійність та безперервність
   - Система повинна автоматично обробляти втрату підключення користувача та дозволяти повторне підключення до сесії.
   - Усі критичні процеси (голосування, хід гри) повинні бути захищені від збоїв та зберігатись на сервері.
3) Безпека
   - Паролі користувачів повинні зберігатися в зашифрованому вигляді.
   - Комунікація між клієнтом і сервером повинна бути захищена (наприклад, через HTTPS і WebSocket Secure).
   - Повинна бути реалізована базова перевірка прав доступу до певних функцій (роль адміністратора, перевірка ходу гри тощо).
4) Зручність та доступність інтерфейсу
   - Інтерфейс повинен бути інтуїтивно зрозумілим для користувачів без попереднього ознайомлення з системою.

- Сайт повинен коректно працювати на більшості сучасних браузерів (Google Chrome, Firefox, Safari, Microsoft Edge).

5) Локалізація
   - Інтерфейс має бути повністю локалізованим українською мовою.
   - Усі системні повідомлення та ігрові дії мають відображатися українською.

1.5.4 Вимоги до UI

Інтерфейс користувача має важливе значення для забезпечення зручності та ефективності взаємодії з системою. Під час формування вимог до дизайну інтерфейсу враховано як загальноприйняті принципи UX/UI, так і особливості жанру гри «Мафія». Основні вимоги сформовано на основі сучасних психологічних законів дизайну та поведінки користувачів.

Основні вимоги до інтерфейсу:

1) Використання темної кольорової гами:
   - Темна тема інтерфейсу відповідає стилістиці гри «Мафія», створює атмосферу інтриги та напруги.
   - Яскраві кольорові акценти (наприклад, помаранчевий для CTA-кнопок) забезпечують візуальну привабливість і привертають увагу.
2) Застосування Закону Якоба (Jakob’s Law) [8]
   - Розташування основних елементів інтерфейсу (навігації, списку гравців, кнопок дій) відповідає типовим очікуванням користувача, що спрощує освоєння інтерфейсу.
   - Хедер, футер, блоки з інформацією розташовані у звичних для більшості сайтів місцях.

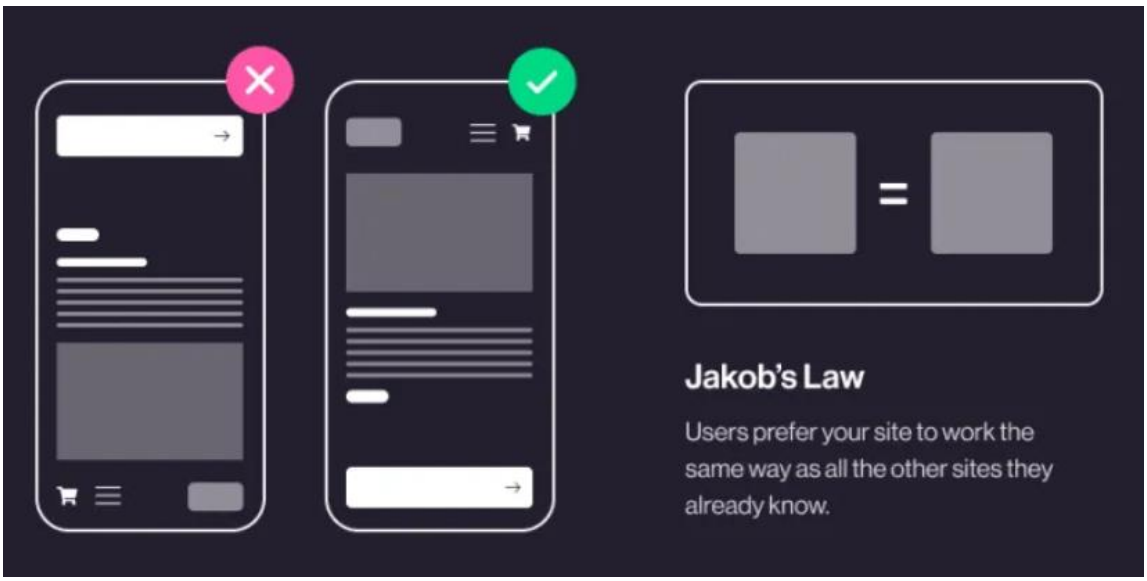

Рисунок 1.4 – Візуалізація Закону Якоба [8]

3) Дотримання Закону Фіттса (Fitts's Law) [9]
    - Всі інтерактивні елементи (кнопки «Почати гру», «Готовий» тощо) мають оптимальні розміри та розташовані на видимих ділянках сторінки, що зменшує час пошуку та взаємодії.
4) Закони Близькості та Загальної Області (Law of Proximity & Common Region) [10]
    - Пов'язані між собою елементи згруповано разом (наприклад, список гравців, налаштування гри).
    - Візуальне обмеження блоків (рамки, тло) допомагає користувачеві краще орієнтуватися в інтерфейсі.
5) Спрощення складних налаштувань гри
    - Оскільки гра «Мафія» може містити багато ролей і варіацій правил, інтерфейс повинен наочно та інтуїтивно показувати етапи налаштування, мінімізуючи когнітивне навантаження.

РОЗДІЛ 2. РОЗРОБКА МОДЕЛІ ТА ПРОГРАМНОГО КОМПЛЕКСУ

2.1 Проєктування веб-застосунку

2.1.1 Визначення архітектури систем

Проаналізувавши дослідження [11], для розробки веб-застосунку було обрано модульну клієнт-серверну архітектуру з підтримкою реального часу, яка реалізована за принципом SPA (Single Page Application) на фронтенді та REST/WebSocket API на бекенді. Такий підхід є оптимальним для багатокористувацьких онлайн-ігор, оскільки дозволяє ефективно розподіляти відповідальність між клієнтською та серверною частинами, забезпечуючи гнучкість, масштабованість та високу продуктивність системи.

Основні компоненти архітектури:

Клієнтська частина(Frontend) реалізована на основі фреймворку React. Це дозволяє створити швидкий, динамічний інтерфейс користувача з використанням компонентного підходу. Реалізація SPA дозволяє уникнути повного перезавантаження сторінки при зміні вмісту, що значно покращує взаємодію користувача з системою.

Серверна частина (Backend) реалізована на основі NestJS, що є фреймворком для Node.js з підтримкою модульної структури, контролерів, сервісів та ін'єкції залежностей. Це дозволяє ефективно організувати бізнес-логіку, обробку запитів та взаємодію з базою даних. Для підтримки живого обміну повідомленнями між клієнтами використовується протокол WebSocket, який дозволяє організувати двостороннє з'єднання між сервером і браузером. Це критично важливо для синхронізації стану гри між усіма гравцями під час сесій.

База даних: Зберігання інформації про гравців, сесії, налаштування гри здійснюється у реляційній базі даних PostgreSQL. Вибір обумовлений її надійністю, розширеними можливостями типізації та хорошою підтримкою у Node.js-середовищі.

Для наочного відображення взаємозв’язків між основними компонентами клієнтської та серверної частин системи, було побудовано діаграму архітектури SPA-додатку (див. рис. 2.1). Вона демонструє загальний принцип функціонування застосунку, де клієнтська частина на React взаємодіє з серверною частиною на базі NestJS через REST API та WebSocket-з’єднання.

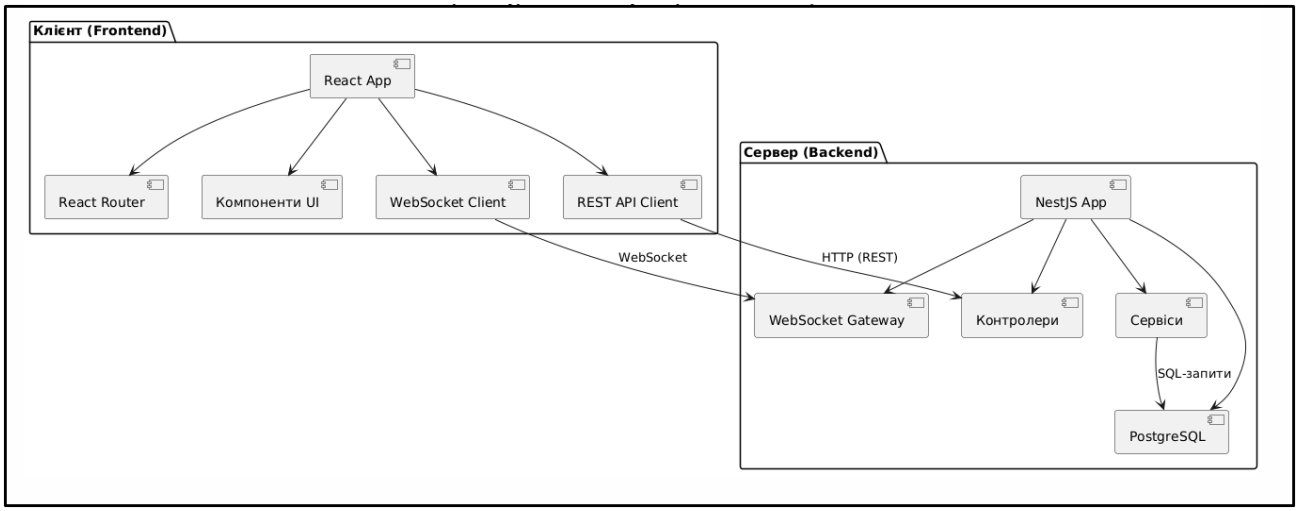

Рисунок 2.1 – Діаграма архітектури системи

2.1.2 Проєктування структури веб-сайту

Одним із ключових етапів під час розробки веб-застосунку є визначення його структури, що забезпечує логічну організацію контенту та зручну навігацію для користувача. Від правильного вибору структури сайту значною мірою залежить зручність користування, масштабованість і загальна ефективність веб-системи.

Проаналізувавши типові підходи до побудови структури веб-сайтів, що мають схожий функціонал (ігрові платформи, сервіси з реєстрацією користувачів, налаштуванням лобі та взаємодією в реальному часі), було обрано деревоподібну структуру сайту. Така модель організації інтерфейсу є найбільш доцільною в умовах великої кількості унікальних сторінок та функціональних блоків. Як

зазначено в [12], деревоподібна архітектура є однією з найефективніших для організації взаємозв’язаного контенту в межах ієрархічної системи.

На діаграмі, зображеній на рисунку 2.2, чітко простежується риса деревоподібної структури — наявність "стовбура дерева", який виконує функцію головного вузла навігації. Від нього відгалужуються всі інші сторінки, що відповідають окремим функціональним модулям.

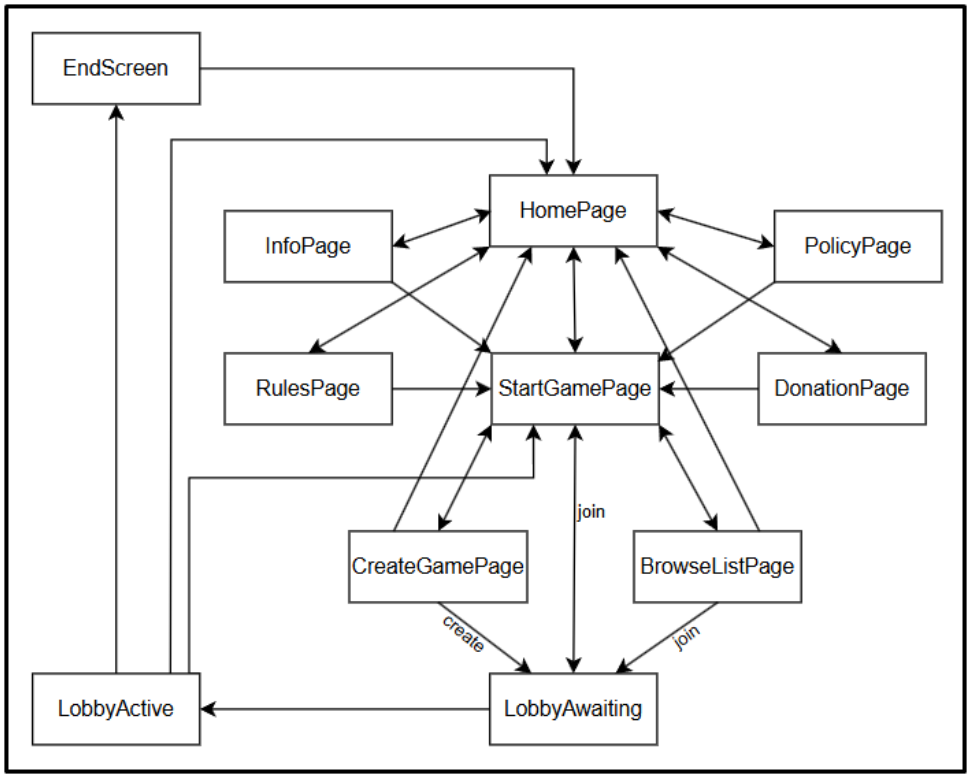

Рисунок 2.2 – Візуальне відображення переходів між сторінками

Кожна гілка (тобто розділ сайту) має логічний шлях повернення до кореневого вузла, що забезпечує зручну та інтуїтивно зрозумілу навігацію. Такий підхід дозволяє з будь-якої частини сайту швидко повернутися до головної сторінки або основного розділу, що є важливою рисою якісного UX-дизайну.

2.1.3 Розробка моделі

Для забезпечення внутрішньої логіки гри «Мафія» було сформовано основні сутності, які відображають структуру ігрового процесу та взаємодію гравців у системі. Кожна сутність описує певний аспект гри та має власні властивості. Нижче наведено перелік ключових сутностей і їхніх характеристик.

Таблиця 2.1 – Основні сутності системи та їх атрибути

Сутність	Атрибути
Game(Гра)	key: string value: Lobby
Player(Гравець)	id?: number; username: string; avatarId: number; role?: string; ready?: boolean; alive?: boolean; action?: boolean;
Role(Роль)	roleName: string; status: boolean; avaible: boolean; action: boolean;
Сутність	Атрибути
Lobby(Лоббі)	host: Player; players: Player[]; settings: Settings; state: GameState; aiAnswer: boolean; nightActions?: NightActions;

Продовження таблиці 2.1

Сутність	Атрибути
GameState(Стан гри)	currentState: PosibleStates; activeRole?: string; readyToVote?: string[]; winner?: string;
NightActions(Нічні дії)	roletTarget: Player

Ця модель є основою для реалізації логіки серверної частини гри та синхронізації стану між клієнтами.

На рисунку В.2 (див. додаток В) зображено діаграму основних сутностей, що використовуються для реалізації логіки гри "Мафія", а також їхні властивості та зв’язки між ними. Кожна сутність представлена у вигляді класу зі списком атрибутів, які відповідають полям у TypeScript-інтерфейсах проєкту.

- Game – загальна точка доступу до поточного екземпляра гри. Містить унікальний ключ та об’єкт типу Lobby, що інкапсулює всі дані гри.
- Lobby – центральна сутність, яка зберігає дані про поточну гру: список гравців (Player[]), налаштування (Settings), поточний стан (GameState), нічні дії (NightActions) та інші допоміжні параметри.
- Player – модель, що представляє окремого гравця, включаючи ім’я, аватар, роль, статуси готовності, життя, голосування тощо.
- Role – роль, яку може мати гравець у грі. Містить атрибути, що описують активність та можливість дії.
- Settings – параметри, задані для гри: кількість гравців, кількість мафії, список доступних ролей.
- GameState – поточний стан гри: етап (ніч, день тощо), активна роль, список гравців, готових до голосування, а також переможець гри.
- NightActions – структура для збереження нічних дій певних ролей, таких як ціль мафії, лікування, блокування тощо.

Зв’язки між класами відображають логічні залежності:

- Game пов’язаний із одним Lobby.
- Lobby містить багато Player та Role, а також поточний GameState і NightActions.
- Кожен Player має одну Role.
- NightActions зберігає посилання на гравців, які стали цілями ролей уночі.

Ця модель дозволяє чітко структуровано представляти логіку та стан гри "Мафія" у вигляді об’єктів, що спрощує реалізацію як фронтенду, так і бекенду застосунку.

2.1.4 Проєктування бази даних

Для забезпечення зберігання, обробки та подальшого аналізу даних, пов’язаних із користувачами, їхньою статистикою та історією ігор, була спроєктована реляційна база даних. При проєктуванні було дотримано вимог третьої нормальної форми (3NF) [13], що дозволяє уникнути надлишкового дублювання даних, зменшує ймовірність логічних помилок при оновленні записів і спрощує масштабування системи в майбутньому [14].

Основними сутностями бази даних стали: гравець, статистика гравця, гра та участь гравця у грі. Також було реалізовано зв’язки між таблицями, що дозволяють визначити, хто брав участь у якій грі, яку роль виконував, та яка сторона здобула перемогу.

Основні вимоги:

- Зберігати користувачів
- Статистику зберігати окремо
- Зберігати історію ігор з коротким описом
- Зв’язки: хто грав у якій грі, яку роль мав, хто виграв

Таблиця 2.2 – Таблиця Players

Поле	Тип	Опис
id	INT(PK)	Унікальний ідентифікатор
username	VARCHAR	Ім’я користувача
Password	VARCHAR	Хеш паролю
Email	VARCHAR	Електронна пошта

Ця таблиця зберігає основну інформацію про користувачів сайту, вона є є центральною для ідентифікації та аутентифікації користувачів.

Таблиця 2.3 – Таблиця PlayerStats

Поле	Тип	Опис
player_id	INT (PK, FK)	Зовнішній ключ на Players(id)
total_games	INT	Кількість зіграних ігор
games_won	INT	Кількість перемог
games_lost	INT	Кількість поразок

Ця таблиця містить персональну статистику кожного гравця та дозволяє швидко отримати статистику для профілю гравця. Має зв’язок 1 – 1 з таблицею Players.

Таблиця 2.4 – Таблиця Games

Поле	Тип	Опис
id	INT(PK)	Унікальний id гри
date_played	DATETIME	Коли відбулась гра
winner_side	VARCHAR	Хто виграв

Суть цієї таблиці у збережені загальної інформації про зіграні ігри, є базою для журналу історії ігор.

Таблиця 2.5 – Таблиця GamePlayers

Поле	Тип	Опис
game_id	INT (PK,FK)	Зовнішній ключ на Games(id)
player_id	INT (PK,FK)	Зовнішній ключ на Players(id)
role	VARCHAR	Роль гравця у грі
alive	BOOLEAN	Чи залишився гравець живим до кінця гри

Містить зв’язки між іграми та гравцями, а також їхні ролі. Д ає змогу зберігати повну історію участі гравців у кожній грі з деталізацією по ролях.

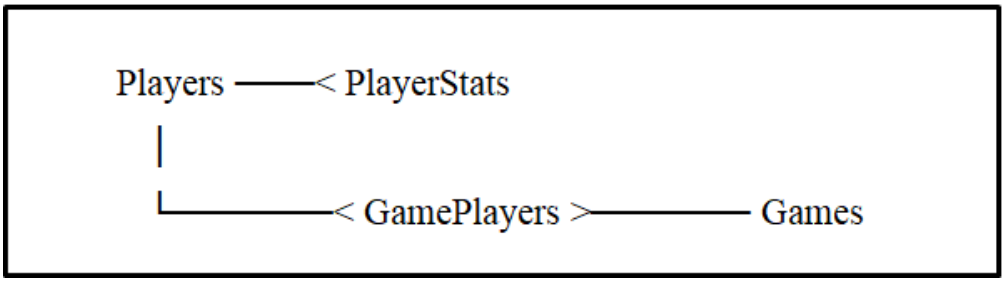

Рисунок 2.3 – ER – модель

У результаті отримано логічно узгоджену структуру, яка забезпечує ефективне управління даними гри “Мафія”.

2.1.5 Розробка бізнес моделі

Основна мета цієї системи — забезпечити зручний і доступний спосіб проведення гри не залежачи від місцезнаходження гравців. Розробка бізнес-моделі дозволяє зрозуміти, для кого створено продукт, які проблеми він вирішує, та які можливості розвитку або масштабування має в подальшому.

Цільова аудиторія: Цільовою аудиторією розробленого веб-додатку є переважно молоді люди віком від 14 до 30 років [15], які мають інтерес до настільних ігор, психологічних ігор на дедукцію, а також до онлайн-розваг у колі

друзів. Це можуть бути: школярі, студенти, учасники спільнот або клубів настільних ігор, кола друзів які перебувають на відстані.

Кожен тип користувачів мають дещо різні очікування від застосунку, проте до головних потреб можна віднести:

- Зручний інтерфейс
- Швидкодія застосунку
- Налаштування ролей та їх кількість
- Доступ з будь-якого пристрою
- Вбудована статистика

Запропонована система вирішує низку актуальних проблем, пов’язаних з проведенням гри “Мафія” у онлайн режимі:

- Відсутність фізичної присутності: У сучасному світі не завжди можливо зібрати всіх гравців офлайн. Система дозволяє організовувати гру дистанційно, об’єднуючи користувачів у спільному лобі через інтернет.
- Труднощі з веденням гри вручну: Традиційна версія гри вимагає ведучого, який контролює правила, ролі, порядок дій, голосування. Система автоматизує цей процес, виключаючи людський фактор і спрощуючи гру.
- Складність у налаштуванні ролей і правил: У багатьох варіантах “Мафії” використовуються нестандартні ролі. Система дозволяє гнучко налаштовувати кількість гравців, типи ролей та їхню поведінку в грі.
- Відсутність статистики та історії ігор: У фізичному форматі складно зберігати дані про результати ігор. У системі ведеться персональна статистика гравця та історія ігор, що дозволяє аналізувати успішність, покращувати навички та змагатися з іншими.
- Складність для новачків: Завдяки інтуїтивному дизайну, покроковій логіці дій і підказкам, система підходить як для досвідчених гравців, так і для новачків.
- Відсутність універсального інструменту: Багато онлайн-альтернатив мають обмежену функціональність або вимагають встановлення додатків.

Запропонована система працює як веб-додаток у браузері, доступний з будь-якого пристрою.

Після аналізу цільової аудиторії та їх основні потреби, було визначено перелік ключових функціональних можливостей системи:

- Створення облікового запису
- Швидке та зручне створення лоббі для гри
- Вибір між ШІ ведучим та ручним
- Налаштування правил гри та кількості активних ролей
- Збереження статистики користувача
- Внутрішньоігровий чат

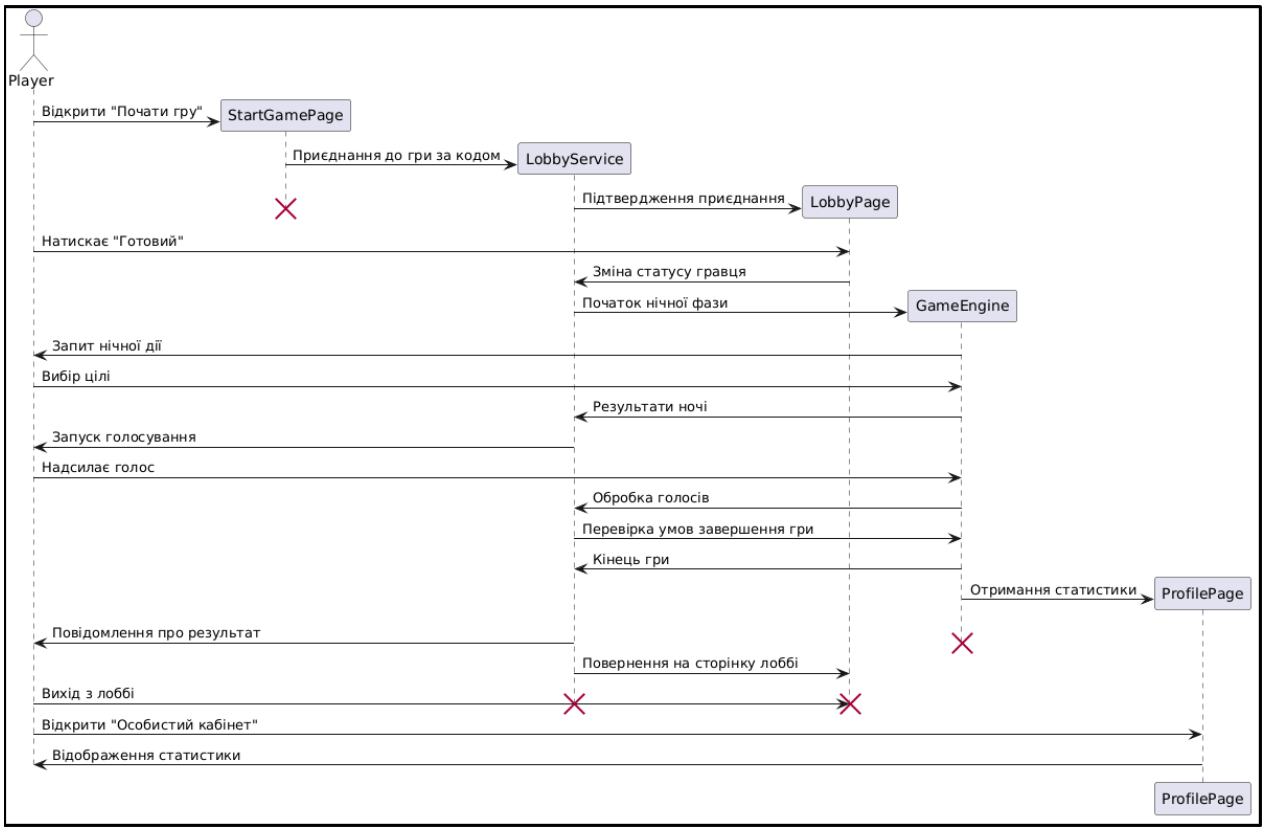

Рисунок 2.5 – Діаграма послідовності головного сценарію гри

На діаграмі, зображеній на рисунку 2.5, описано повний життєвий цикл взаємодії гравця з веб-застосунком — від початку сесії гри до перегляду особистої статистики. Сценарій включає етапи підключення до лобі за кодом, встановлення статусу "готовий", виконання нічних дій, участь у голосуванні, завершення гри та

вихід з лобі. Можна помітити моменти створення і знищення ігрового рушія (GameEngine), який активується на початку нічної фази та припиняє своє існування після завершення гри. Завдяки цьому забезпечується ефективне використання ресурсів сервера та логічне розділення ігрових сесій. У фіналі сценарію гравець має змогу переглянути зібрану статистику у своєму профілі.

2.1.6 Проєктування архітектури системи

Архітектура створюваного веб-додатку базується на класичній клієнт-серверній моделі, що дозволяє чітко розділити відповідальність між фронтендом і бекендом. Клієнтська частина відповідає за взаємодію з користувачем, відображення інтерфейсу та передачу запитів до сервера, тоді як серверна частина реалізує бізнес-логіку гри, забезпечує зв’язок між гравцями в реальному часі та керує роботою бази даних.

Однією з ключових ідей при структуризації коду було суворе дотримання принципу розділення обов’язків. Кожна частина застосунку — чи то візуальний компонент, логіка взаємодії з сервером, чи обробка станів гри — виконує лише свою вузьку функцію. Завдяки цьому система залишається зрозумілою, легкою для підтримки та розширення. Це особливо важливо в контексті багатокористувацької гри, де велика кількість подій відбувається одночасно.

Особливу увагу приділено модульності: як у клієнтській, так і в серверній частині, кодова база розбита на ізольовані частини — модулі, кожен з яких відповідає за окрему функціональність, наприклад, гру, чат, профіль або статистику. У серверному коді, побудованому з використанням NestJS, це реалізується через декларативну систему модулів, тоді як у клієнтському — через розділення на компоненти React.

Ще одним важливим принципом є відокремлення бізнес-логіки на стороні сервера. Усі ключові дії гри, зокрема обробка фаз ночі та голосування,

виконуються виключно на бекенді, що гарантує контроль над перебігом гри та захист від стороннього втручання.

Взаємодія між клієнтом і сервером реалізована через REST API та WebSocket. API використовується для обробки стандартних запитів — реєстрації, входу, перегляду статистики — тоді як WebSocket забезпечує обмін подіями в реальному часі під час гри [16].

2.1.6.1 Архітектура клієнтської частини:

Клієнтська частина веб-застосунку реалізована за допомогою бібліотеки React, яка дозволяє створювати гнучкий і масштабований інтерфейс користувача. Архітектура побудована відповідно до компонентного підходу, де кожен елемент сторінки — окрема одиниця з власним станом, логікою та відображенням. Це забезпечує повторне використання коду та його логічне групування за функціональністю.

В основі структури клієнта лежить поділ на кілька основних директорій. Каталог components містить повторно використовувані UI-компоненти, такі як кнопки, модальні вікна, поля введення.

У папці view розміщено компоненти, що відповідають окремим сторінкам додатку — головна, лобі, гра, профіль тощо. Навігація між ними реалізована за допомогою бібліотеки React Router, що забезпечує зручний механізм маршрутизації.

Логіка взаємодії з сервером винесена в окрему директорію api де визначено всі запити до REST API та підключення до WebSocket. Це дозволяє розмежувати візуальну частину інтерфейсу від роботи з даними, що відповідає принципам «чистої архітектури» [17].

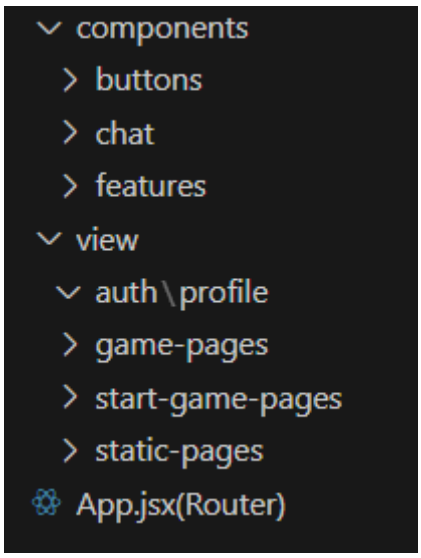

Рисунок 2.6 – Структура розміщення сторінок та компонентів

Для зберігання стану застосунку використано як локальний стан компонентів, так і глобальний — через контексти (React Context). Наприклад, контекст автентифікації дозволяє перевіряти, чи залогінений користувач, незалежно від того, на якій сторінці він перебуває. Також присутній окремий контекст гри, який зберігає стан лобі, гравців, хід гри та поточну фазу.

Інтерактивність та швидка реакція на дії користувача досягається завдяки використанню хуків (useState, useEffect, useContext тощо), що дозволяють обробляти зміни стану та виконувати побічні ефекти, наприклад, підключення до WebSocket або оновлення інтерфейсу при зміні фаз гри.

Такий підхід забезпечує як простоту розуміння структури застосунку, так і його розширюваність. У разі необхідності додавання нових сторінок або розширення функціоналу, достатньо створити окремі компоненти та підключити їх до існуючої логіки.

2.1.6.2 Архітектура серверної частини:

Серверна частина вебдодатку реалізована за допомогою NestJS — прогресивного фреймворку для Node.js, який базується на TypeScript та використовує архітектурний підхід Modular + MVC + Dependency Injection, що

дозволяє гнучко масштабувати систему, забезпечувати високий рівень розділення відповідальностей та підтримувати чисту структуру коду.

Архітектура серверної частини організована у вигляді модулів, де кожен модуль відповідає за окрему функціональність або підсистему. Такий підхід дозволяє легко ізолювати логіку, повторно використовувати її та розширювати проєкт без порушення існуючих залежностей.

У проєкті виділено такі основні модулі:

- AuthModule – обробка реєстрації, авторизації, генерація JWT-токенів та хешування паролів
- LobbyModule – логіка управління лобі, створення, приєднання, збереження стану гравців та налаштувань
- GameModule – обробка нічної фази, голосувань, завершення гри та визначення переможця
- PlayerModule – взаємодія з профілями гравців, статистикою, історією ігор.
- DatabaseModule – підключення до бази даних

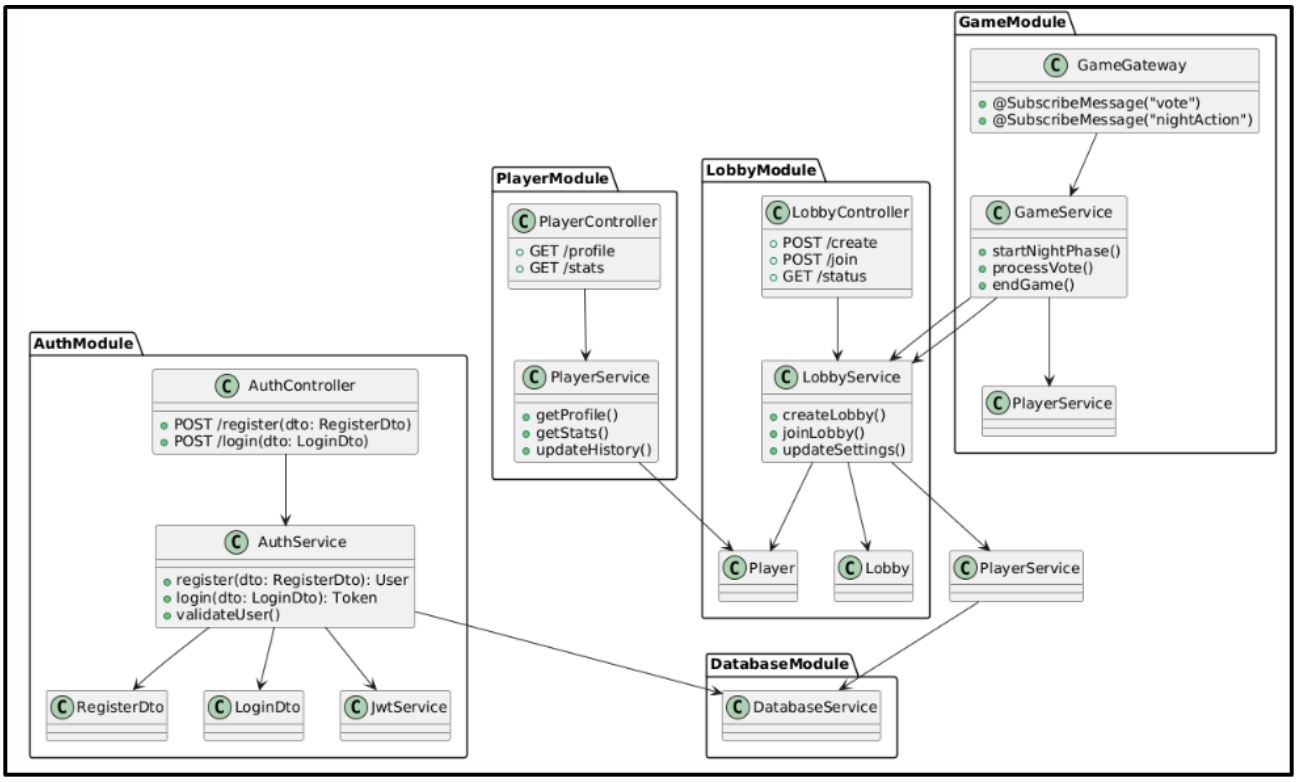

Рисунок 2.7 – Діаграма модулів

Для спрощеного розуміння архітектури серверної частини веб-додатку було створено діаграму модудів (див. рис. 2.7), на якій зображено основні зв’язки між модулями системи.

Діаграма відображає модульну структуру застосунку, де кожен модуль відповідає за окрему функціональність — авторизацію, управління лобі, ігрову логіку, обробку профілю гравця та взаємодію з базою даних. Вказано основні сервіси, контролери, об'єкти передачі даних (DTO), а також взаємозв’язки між ними, що забезпечують узгоджену взаємодію компонентів та підтримку принципів розділення відповідальностей.

Всі модулі взаємодіють між собою через сервіси, які відповідають за бізнес-логіку, тоді як контролери обробляють HTTP-запити або WebSocket-події. В якості обміну даними використовуються DTO-об'єкти, які проходять перевірку (validation) перед обробкою.

З’єднання клієнта з сервером відбувається через WebSocket, що дозволяє реалізувати події у реальному часі наприклад, повідомлення в чаті, нічні дії, голосування тощо.

Таким чином, серверна частина спроєктована як набір незалежних, ізольованих модулів із чіткими обов’язками, що дозволяє досягти високої гнучкості, надійності та масштабованості.

2.2 Конструювання веб-застосунку

На цьому етапі розробки було здійснено безпосереднє створення програмного забезпечення відповідно до спроєктованої архітектури, моделей даних та функціональних вимог. Конструювання охоплює реалізацію як серверної, так і клієнтської частин системи, розробку користувацького інтерфейсу, налаштування системи контролю версій та виконання тестування на відповідність заданим вимогам.

Розробка серверної частини передбачає створення основних модулів, які реалізують бізнес-логіку гри, обробку запитів, взаємодію з базою даних, авторизацію користувачів та управління сесіями гри. Архітектурні принципи модульності, розділення відповідальностей (MVC) та впровадження залежностей (Dependency Injection) забезпечують масштабованість та підтримуваність коду.

У клієнтській частині створено компоненти, які відповідають за навігацію, обробку подій від користувача та візуалізацію даних. Графічний інтерфейс було реалізовано з урахуванням принципів UX/UI дизайну для досягнення інтуїтивної та зручної взаємодії з додатком.

2.2.1 Реалізація основних модулів серверної частини

Основним модулем серверної частини системи є LobbyModule, реалізований у файлі lobby.module.ts. Цей модуль відіграє ключову роль в організації та керуванні ігровими лобі, забезпечуючи основний функціонал для взаємодії користувачів із сервером.

Лістинг модуля:

```
@Module({
  providers: [LobbyService, LobbyGateway, RoleActionService],
  controllers: [LobbyController],
  exports: [LobbyService, LobbyGateway, RoleActionService],
})
```

2.2.1.1 LobbyService

Сервіс LobbyService є центральним компонентом модулю лобі і відповідає за реалізацію бізнес-логіки, пов’язаної з керуванням ігровими лобі, учасниками та ігровим процесом. Він інкапсулює функціонал створення, оновлення та контролю стану лобі, а також забезпечує взаємодію між користувачами через WebSocket-з’єднання, реалізоване у LobbyGateway.

Основною структурою даних, яку оперує сервіс, є об’єкт лобі, що містить інформацію про хоста, список гравців, ігрові налаштування, поточний стан гри та дії, виконувані під час нічної фази.

Лістинг вигляду об’єкта:

```
export interface Lobby {
  host: Player;
  players: Player[];
  settings: Settings;
  state: GameState;
  aiAnswer: boolean;
  nightActions?: NightActions;
}
```

Кожен гравець моделюється як об’єкт з властивостями, що визначають унікальний ідентифікатор, ім’я, роль, стан готовності та життєздатності.

Лістинг об’єкту Player:

```
export interface Player {
  id?: number;
  username: string;
  avatarId: number;
  role?: string;
  ready?: boolean;
  alive?: boolean;
  action?: boolean;
  votes?: number;
  voted?: boolean;
}
```

Метод createLobby відповідає за ініціалізацію нового ігрового лобі, створюючи унікальний ідентифікатор, реєструючи хоста та налаштовуючи параметри гри, такі як кількість гравців, ролі та штучний інтелект для режиму без ведучого. Лістинг наведено у додатку А.1 (див. додаток А)

Приєднання гравців до лобі обробляється методом joinLobby, який перевіряє наявність вільних місць та існування лобі, додаючи нового гравця до списку учасників.

Лістинг методу:

```
joinLobby(lobbyId: string, player: Player) {
  const lobby = this.lobbies.get(lobbyId);
  if (!lobby) {
    return { success: false, message: 'Лобі не знайдено' };}
  if (lobby.settings.playersNumber === lobby.players.length) {
    return { success: false, message: 'Лоббі заповнене' };}
```

```
 lobby.players.push(player);
 player.id = lobby.players.length;
return { success: true, players: lobby.players, id: player.id };}
```

Для отримання інформації про лобі передбачені методи getLobby та getFullLobby, які повертають дані з різним рівнем деталізації, приховуючи або розкриваючи рольову інформацію відповідно.

Лістинг getFullLobby:

```
getFullLobby(lobbyId: string) {
  const lobby = this.lobbies.get(lobbyId);
  if (!lobby) return { message: 'Лобі не знайдено' };
  return lobby;
}
```

Усі вище наведені лістинги наведені з ціллю продемонструвати зручність взаємодії із головною сутністю “Гра” завдяки використанню асоціативного масиву Map, де ключем виступає lobbyId, а значенням об’єкт типу Lobby:

```
private lobbies = new Map<string, Lobby>();
```

Такий підхід забезпечує ефективний доступ, зміну та керування ігровими сесіями за унікальним ідентифікатором лобі, що суттєво спрощує реалізацію логіки гри.

Важливою частиною ігрового процесу є нічна фаза, що виконується асинхронно в методі performNightPhase. Він по черзі активує ролі, які мають діяти вночі, очікуючи на їхні дії, і в кінці підсумовує результати, змінюючи стан гри на "день" або "завершено" відповідно до умови перемоги.

Для обробки конкретних дій гравців під час нічної фази реалізовано метод performRoleAction, який делегує виконання ролей сервісу RoleActionService залежно від ролі гравця, наприклад, дії мафії, лікаря чи коханки та в якому реалізовано асинхронне очікування дії від гравців за допомогою методу waitForPlayerActions. Лістинги методів наведені у додатку А.2 (див. додаток А)

З метою наочної та зрозумілої демонстрації логіки роботи основних методів, які відповідають за нічну фазу гри, було створено відповідну блок-схему, що ілюструє послідовність виконання ключових кроків, перевірок та умов, які виникають у процесі виконання функціоналу (див. додаток В.1).

Таким чином, LobbyService реалізує комплексний набір методів для повного контролю життєвого циклу лобі — від створення та формування гравців до завершення гри з визначенням переможців, забезпечуючи при цьому інтерактивність і узгодженість станів через механізми обміну повідомленнями.

2.2.1.2 LobbyController

LobbyController виконує роль HTTP-контролера у архітектурі NestJS і є посередником між клієнтськими запитами та бізнес-логікою, реалізованою у LobbyService. Він відповідає за маршрутизацію запитів, пов’язаних із життєвим циклом ігрового лобі: від створення та приєднання гравців — до запуску гри та обробки дій під час неї.

Контролер взаємодіє з LobbyGateway, щоб оперативно повідомляти клієнтів про зміни у стані гри або лобі наприклад, при розподілі ролей.

Лістинг:

```
@Post(':lobbyId/assign-roles')
  assignRoles(@Param('lobbyId') lobbyId: string) {
    this.lobbyService.assignRoles(lobbyId);
    this.lobbyGateway.emitLobbyPlayers(lobbyId);
    return { message: 'Ролі розподілені' };
  }
```

Завдяки чіткому розділенню відповідальностей, контролер залишається легким і зручним для тестування, водночас забезпечуючи повну функціональність взаємодії клієнта з грою.

2.2.1.3 LobbyGateway

LobbyGateway реалізує серверну частину комунікації по протоколу WebSocket. Його основною задачею є забезпечення обміну повідомленнями в реальному часі між сервером і клієнтами. Він дозволяє оперативно передавати інформацію про зміни в стані гри, діях гравців, внутрішньому стані самої кімнати, а також повідомлення у чаті [18].

Для реалізації комунікації використовується бібліотека socket.io, яка дозволяє зручно транслювати події на клієнти, об’єднані в так звані “кімнати” на основі lobbyId. У такий спосіб події транслюються лише тим користувачам, які приєднані до конкретного лобі.

Одним з ключових методів є emitLobbyPlayers, який відповідає за надсилання поточного списку гравців у лобі до всіх підключених клієнтів. Цей метод викликається щоразу, коли хтось приєднується до лобі, змінює статус готовності або отримує роль. Для отримання актуальних даних використовується `lobbyService.getFullLobby(lobbyId)`, що дозволяє витягнути повний об'єкт лобі з внутрішньої пам'яті.

Лістинг:

```
  emitLobbyPlayers(lobbyId: string) {
    const lobby = this.lobbyService.getFullLobby(lobbyId);
    if ('players' in lobby) {
      this.server.to(lobbyId).emit('lobbyUpdated',
lobby.players);
    } else {
      console.error(`Лобі ${lobbyId} не знайдено.`);
    }
  }
```

Цей метод викликається за допомогою handleJoinLobby яка обробляє приєднання клієнта до лобі через WebSocket завдяки декоратору `@SubscribeMessage('joinLobby')`. Вона викликає метод socketsJoin, який додає сокет клієнта до кімнати, і відразу після цього викликає `emitLobbyPlayers`, щоб передати новому користувачу список гравців.

HandleChatMessage, наступний метод який використовує декоратор, дає змогу надсилати повідомлення в загальний чат. Сервер транслює такі повідомлення в кімнату, і всі гравці бачать новий запис у своєму чаті, включаючи ім’я відправника.

Лістинг:

```
 @SubscribeMessage('chatMessage')
  handleChatMessage(
    @MessageBody() data: { lobbyId: string; username: string;
text: string },
```

```
  ) {
    this.server.to(data.lobbyId).emit('chatMessage', {
      username: data.username,
      text: data.text,
    });
  }
```

В майбутньому завдяки цьому методу будуть реалізовані сповіщення від системи наприклад в методі startGame після успішного початку гри

Лістинг:

```
this.lobbyGateway.server.to(lobbyId).emit('chatMessage', {
  username: 'Система',
  text: 'Гра розпочалася!',
});
```

Таким чином, LobbyGateway не зберігає стану гри або гравців самостійно – він виступає лише передавачем інформації між клієнтами та основною логікою гри, що реалізована в LobbyService. Завдяки цьому досягається чітке розділення відповідальностей між бізнес-логікою гри та комунікаційним шаром.

Цей підхід дозволяє масштабувати архітектуру проєкту, підключати нові типи повідомлень і забезпечити стабільну взаємодію користувачів із грою без перезавантажень сторінки або ручного опитування сервера.

Таким чином, LobbyModule є центральним елементом серверної архітектури, що інтегрує основні функціональні можливості для ефективної організації ігрового процесу та комунікації між гравцями.

2.2.2 Реалізація основних компонентів клієнтської частин

Клієнтська частина системи реалізована з використанням бібліотеки React, що дозволяє будувати компонентно-орієнтований інтерфейс із можливістю динамічного оновлення контенту. Вона є відповідальною за відображення інтерфейсу користувача, організацію взаємодії з сервером через REST API та

WebSocket, а також обробку введення користувача та оновлення стану застосунку у відповідь на зміну даних.

2.2.2.1 Взаємодія із сервером

Лістинг socket.js що встановлює з’єднання із сервером:

```
import { io } from "socket.io-client";
export const socket = io("http://localhost:3000", {
  transports: ["websocket"],
});
```

Компоненти, такі як ChatModal, підписуються на події.

Лістинг:

```
useEffect(() => {
  socket.emit("joinChat", lobbyId);
  socket.on("chatMessage", handleNewMessage);
  return () => socket.off("chatMessage", handleNewMessage);
}, [lobbyId]);
```

Для взаємодії з REST-інтерфейсом серверної частини використовується бібліотека axios.

Лістинг створення лобі:

```
const createLobby = async () => {
  const response = await fetch("/api/lobby", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ host: username }),
  });
  const data = await response.json();
  navigate(`/lobby/${data.lobbyId}`);
};
```

2.2.2.2 Головні компоненти інтерфейсу

Маршрутизація в клієнтській частині реалізована за допомогою бібліотеки react-router-dom. Це дозволяє логічно структурувати переходи між різними сторінками у межах одного HTML-документа без необхідності перезавантаження всієї сторінки. Основну конфігурацію маршрутизації наведено в додатку А.3 (див. додаток А)

Така конфігурація забезпечує перехід між головною сторінкою, формами створення та під'єднання до гри, лобі, самою грою, фінальною сторінкою, а також інформаційними підрозділами.

Уся клієнтська частина побудована на базі компонентів — окремих функціональних блоків, що мають свій стан і відображення. Наприклад, компонент ChatModal відповідає за логіку і рендер чату в реальному часі. Він містить обробники WebSocket-повідомлень, механізм додавання повідомлень, а також контролі для вводу повідомлень. Лістинг компонента наведено у додатку А.4 (див. додаток А)

Цей компонент інтегрується із сервером за допомогою WebSocket-з'єднання, що дозволяє забезпечити миттєву передачу повідомлень.

Щодо роботи зі станами, клієнтська частина отримує оновленні стани гри за допомогою API. REST API використовується на етапах створення лобі, отримання списку доступних ігор, перевірки статусу гравців тощо. Для більш динамічної комунікації в процесі самої гри застосовується WebSocket-протокол. Наприклад, коли гравець виконує дію ролі або відправляє голос під час голосування, відповідні повідомлення передаються через WebSocket [19].

Компонент LobbyAwainting є центральною точкою обробки станів лобі в клієнтській частині гри. Його основна роль — координувати відображення відповідних підкомпонентів на основі поточного стану гри, підтримувати синхронізацію даних з сервером через WebSocket та REST-запити, а також налаштовувати поведінку гравця відповідно до його ролі (гравець або ведучий).

Компонент LobbyAwainting відповідає за логіку очікування, підготовки до гри, проведення гри (ніч, день, голосування), завершення гри та повернення до

перегляду результатів. Залежно від стану (state.currentState) він динамічно рендерить відповідний компонент інтерфейсу.

Компонент під'єднується до серверу через WebSocket (socket.emit, socket.on) і через HTTP-запити (axios.get, axios.post, axios.patch). Він реагує на три основні події:

- lobbyUpdated — оновлення списку гравців;
- stateUpdated — зміна загального стану гри (наприклад, перехід до голосування);
- lobbyFullUpdate — повне оновлення лобі, включаючи список гравців, стан гри, нічні дії.

Всю цю інформацію LobbyAwainting розподіляє між дочірніми компонентами за допомогою пропсів (settings, host, users, state, nightActions, aiAnswer тощо). Наприклад компонент EndScreen лістинг якого наведено у додатку А.5 (див. додаток А) отримує інформацію про результат гри, ролі гравців і візуалізує її відповідно до правил гри. Ролі, що були активними, виводяться окремо, а всі інші — групуються як мирні.

2.2.3 Розробка GUI

Інтерфейс користувача є одним із ключових елементів вебзастосунку, оскільки саме через нього здійснюється вся взаємодія гравця з системою. На цьому етапі проєктування і реалізації була поставлена мета створити простий, інтуїтивно зрозумілий та адаптивний інтерфейс, який не лише відображає основну логіку гри, але й допомагає користувачу орієнтуватися у складних ігрових ситуаціях — таких як нічні дії, голосування чи завершення гри.

Під час реалізації графічного інтерфейсу було дотримано сучасних підходів до побудови UI на основі компонентної архітектури. Кожна сторінка або логічна одиниця (лобі, гра, чат, статистика) реалізована у вигляді окремого компоненту,

який відповідає за відображення і взаємодію в межах конкретної підфази гри. Завдяки цьому інтерфейс залишається масштабованим і зручним у підтримці.

Попередньо у Figma було створено повний макет веб-застосунку, що охоплює всі основні сторінки гри: головну, лобі, екран гри, сторінку правил, статистику та інші. Дизайн мав на меті забезпечити логічну структуру, зручну навігацію та візуальну привабливість.

Візуальні прототипи слугували основою для подальшої реалізації інтерфейсу на клієнті. Усі елементи інтерфейсу було адаптовано до компонентної архітектури React, що дало змогу розділити систему на окремі, незалежні та повторно використовувані частини. Це спростило структуру коду, дозволило легко оновлювати окремі частини інтерфейсу та підтримувати зручність у масштабуванні.

Кожна сторінка або підстан гри реалізована у вигляді окремого компонента, який відповідає за свій контекстний сценарій. Гнучкий дизайн дозволив зробити застосунок адаптивним — інтерфейс коректно відображається як на комп’ютерах, так і на мобільних пристроях.

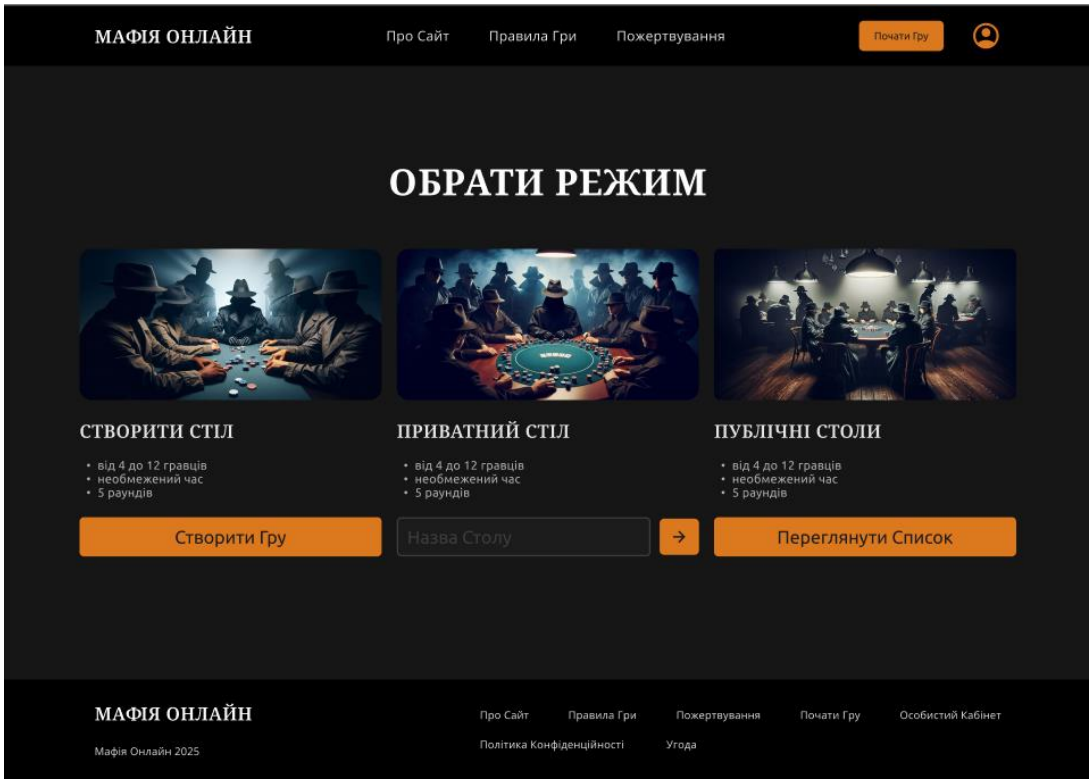

Рисунок 2.9 – Сторінка “Почати Гру”

Сторінка “Почати Гру” разом із головною сторінкою виконує роль центрального вузла — своєрідного "стовбура" у структурі навігаційного дерева веб-застосунку. Їх структура, доступність і логіка переходів мають критичне значення для загального користувацького досвіду.

На всіх подальших макетах інтерфейсу можна спостерігати наявність постійного елементу в шапці — кнопки "Почати гру", яка виконує функцію навігаційного маркера та забезпечує швидкий доступ до сторінки запуску гри. Таким чином, вона слугує основною точкою входу до ігрової логіки застосунку.

Основне призначення цієї сторінки полягає в тому, щоб надати користувачу повний набір засобів для старту сесії: створення ігрового лобі, можливість приєднатися до вже створеного лобі за допомогою унікального коду, а також доступ до списку відкритих лобі, які очікують на нових учасників.

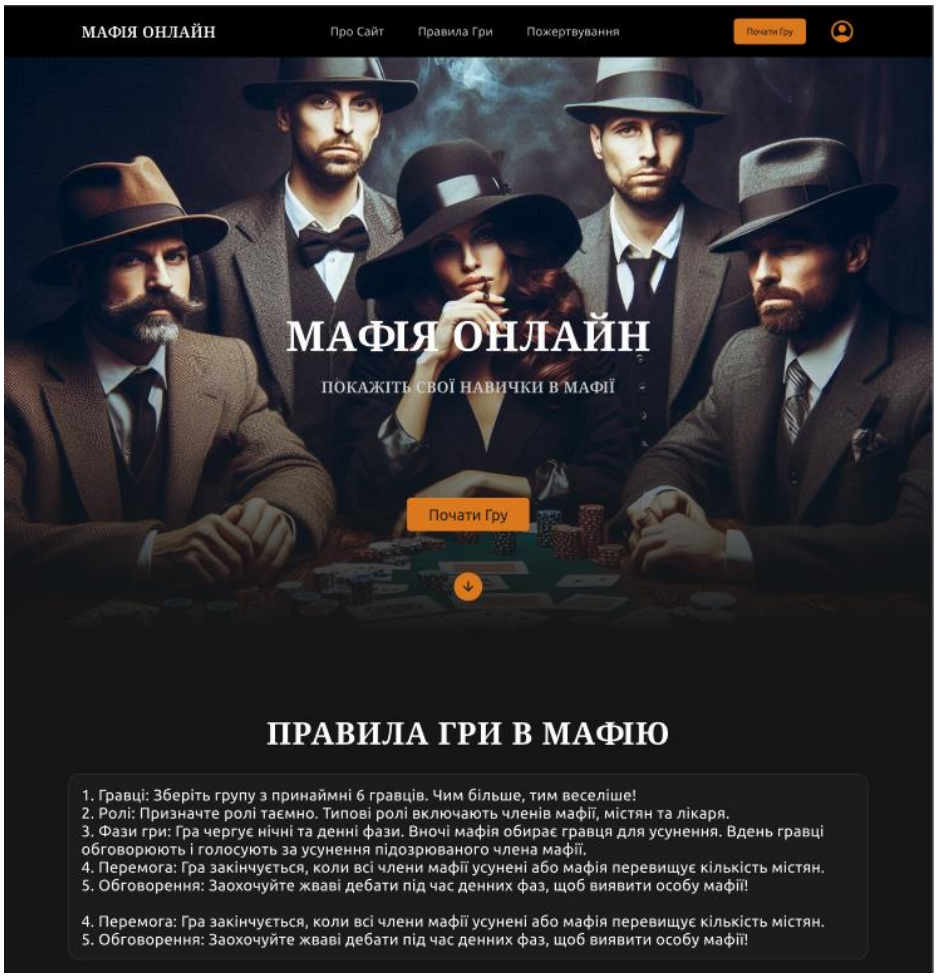

Рисунок 2.10 – Вигляд головної сторінки

Головна сторінка є початковою точкою взаємодії користувача із веб-застосунком, саме вона першою відображається при переході за основним посиланням. Її структура орієнтована як на нових, так і на досвідчених користувачів. Для перших вона виконує ознайомчу функцію, надаючи доступ до правил гри, пояснень щодо принципів роботи системи та особливостей участі в онлайн сесіях. Для користувачів, які вже знайомі з функціоналом, головна сторінка слугує навігаційним центром, своєрідним "коренем дерева", від якого відгалужуються всі основні маршрути переходу до ключових сторінок застосунку.

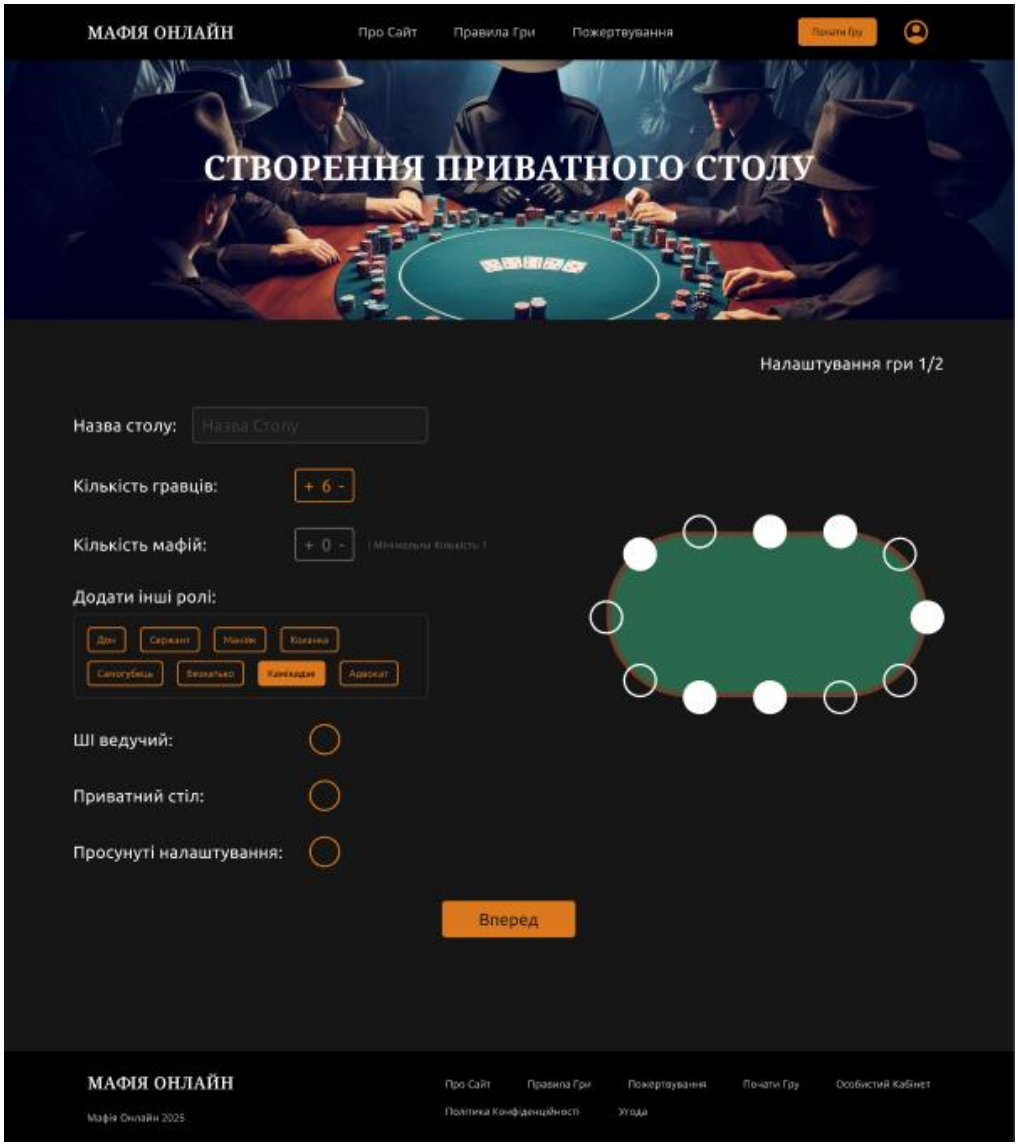

Рисунок 2.11 – Сторінка створення кімнати

Мета сторінки “Створення кімнати” полягає у наданні користувачеві інтуїтивного та ефективного інтерфейсу для налаштування параметрів ігрового лобі. Серед основних опцій, доступних на цій сторінці: визначення кількості учасників, кількості мафій, вибір ролей, які братимуть участь у грі, а також конфігурація режиму ведучого (ручний або автоматичний). З метою підвищення візуальної привабливості та забезпечення кращої орієнтації, інтерфейс доповнено графічним елементом: круглим столом, що динамічно відображає кількість обраних гравців і підтримує загальну естетику дизайну сайту.

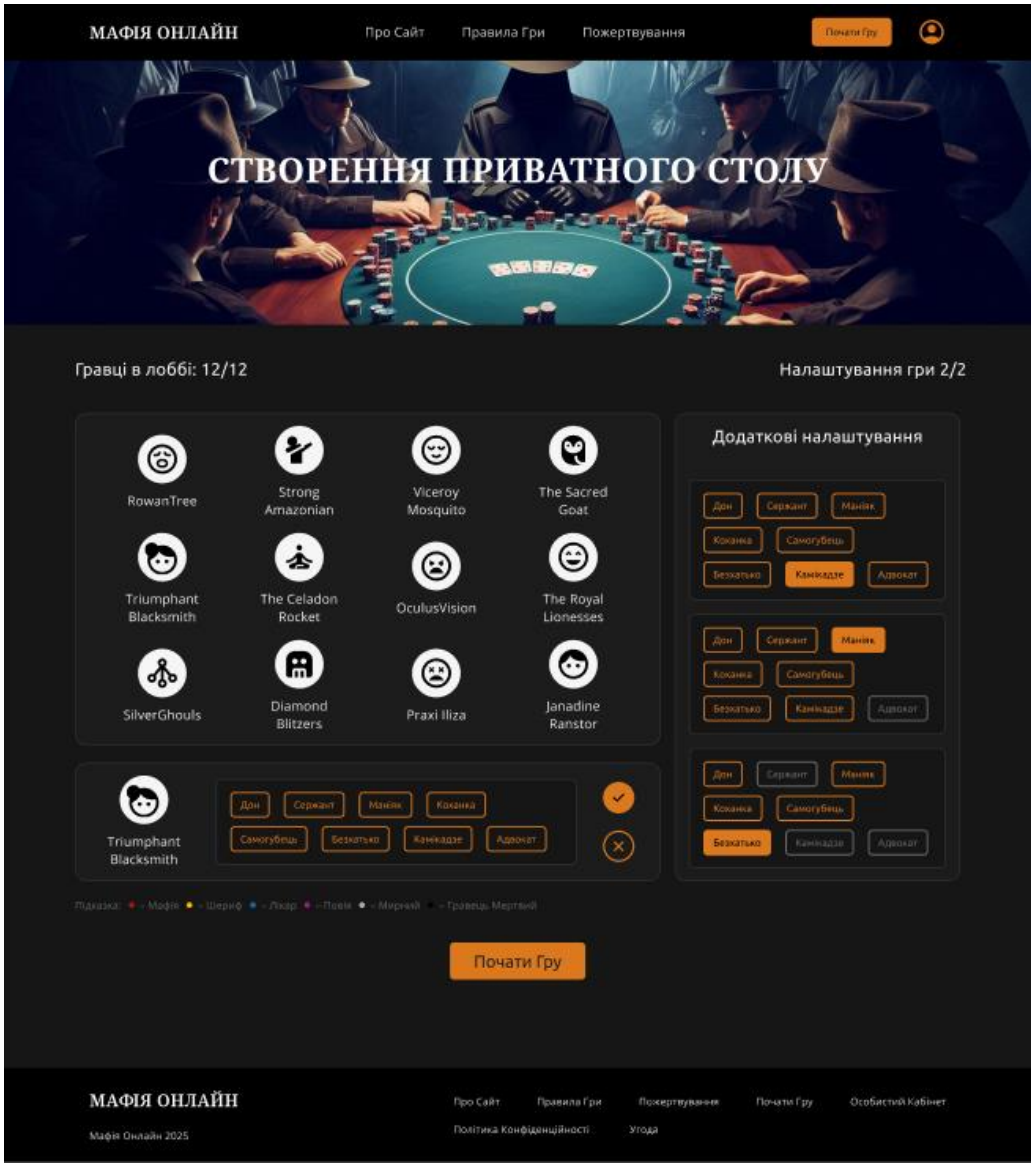

Рисунок 2.12 – Сторінка лоббі при стані користувача Ведучий

Після створення кімнати користувач отримує роль ведучого, а сторінка змінюється у сторінку лобі із станом “очікування”, де реалізовано функціональність для подальшого керування ігровим лобі. У цьому режимі ведучому надається доступ до таких інструментів, як: призначення ролей гравцям, можливість виключення користувачів з лобі, зміна налаштувань гри та ініціація її початку. Окрім панелі керування, сторінка містить список усіх гравців, які приєдналися до лобі. У разі, якщо було обрано режим “Автоматичний ведучий”, сам ведучий також буде відображений у цьому списку як учасник гри.

Рисунок 2.13 – Сторінка лоббі при стані користувача Гравець

Після приєднання до кімнаті за кодом чи за допомогою списку кімнат, користувач набуває стану Гравець. Сторінка лобі при стані очікування для гравця та ведучого дуже відрізняються. Для гравця заборонені інструменти керування

лобі. Йому доступно інструменти взаємодії із чатом та зміна свого стану на “Готовий” та навпаки. Додатково на сторінці розміщено налаштування які задав ведучий та унікальний код кімнати із зручним інструментом копіювання

Рисунок 2.14 – Сторінка гри при стані користувача Ведучий

Після того як ведучий ініціює початок гри, сторінка лобі переходить у стан активної сесії, яка охоплює послідовні фази: ніч, день, голосування та завершення гри. Для ведучого інтерфейс у цьому режимі доповнюється спеціалізованими

інструментами та компонентами, які забезпечують контроль за перебігом гри. Зокрема, йому доступний системний чат, у якому фіксуються ключові дії гравців (наприклад, нічні вибори цілей чи результати голосування). Окрім цього ведучий може ініціювати приватний чат з окремими гравцями або ролями, не обмежуючись лише загальним чатом.

Рисунок 2.15 – Сторінка гри при стані користувача Гравець

У період активної гри гравець взаємодіє з інтерфейсом переважно через контекстну кнопку, розташовану в нижньому меню сторінки. Її функціональність

динамічно змінюється залежно від поточного етапу гри, статусу гравця (наприклад, живий чи мертвий) та його ролі. У нічну фазу ця кнопка може виконувати роль підтвердження дії (вибору цілі), а в інші — слугувати для голосування чи переходу до наступного етапу.

Для парних ролей, таких як Дон і Мафія, реалізовано механізм ідентифікації союзників. Відповідні гравці отримують візуальне позначення ролей своїх товаришів, а також чат ролей автоматично інформує таких гравців про учасників їхньої фракції, забезпечуючи повноту контексту для прийняття стратегічних рішень.

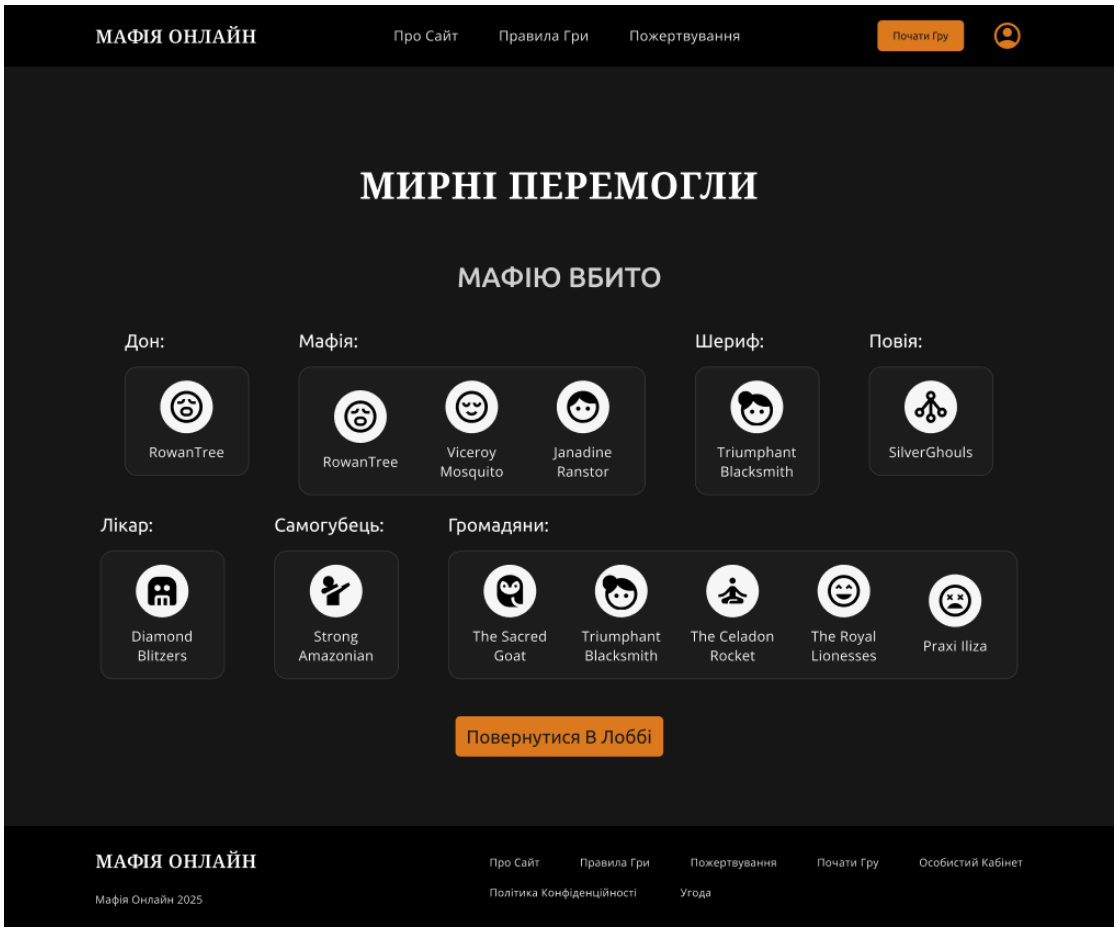

Рисунок 2.16 – Сторінка завершення гри

Завершальний стан лобі є спільним як для гравців, так і для ведучого. У цьому режимі інтерфейс відображає підсумки гри, оголошується переможна сторона, кожному користувачу демонструються ролі інших учасників. Додатково

передбачено кнопку, що дозволяє ініціювати повторний запуск гри, повернувши лобі до стану очікування, що зручно для серійного проведення ігор без необхідності створювати нову кімнату.

2.3 Робота із Git

GitHub виступав основним інструментом для зберігання, відстеження та координації змін у кодовій базі. Завдяки йому забезпечувався стабільний робочий процес та зберігалася історія розробки, що дозволяло повертатися до попередніх версій при потребі.

Основу репозиторію [20] становить чітка структура, поділена на дві основні директорії: client та server, що відповідають клієнтській та серверній частинам веб-застосунку відповідно.

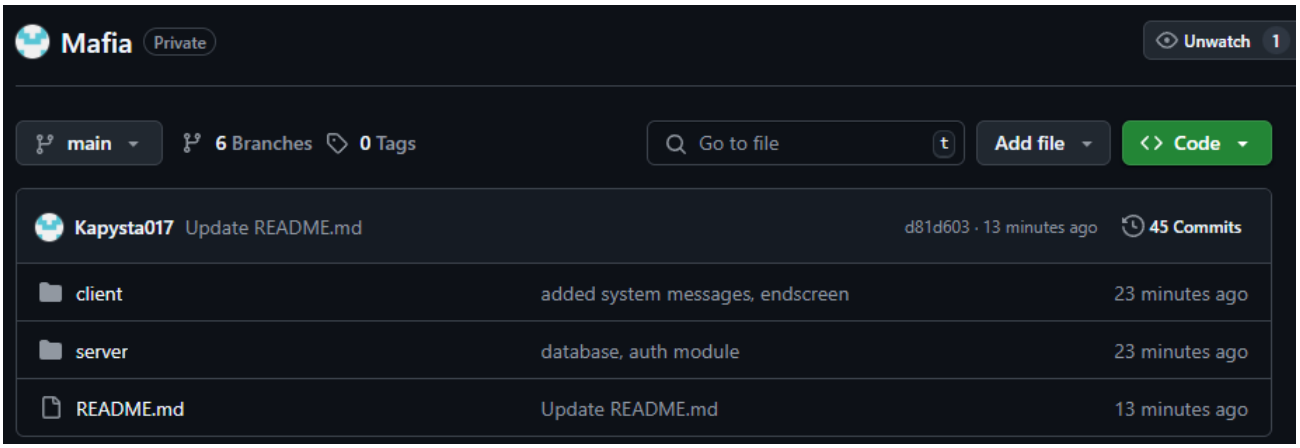

Рисунок 2.17 – Вигляд репозиторію у Github.com

У процесі роботи коміти створювалися за допомогою вбудованого Git-інструментарію в середовищі Visual Studio Code, що забезпечувало зручний контроль версій. Назви комітів формувалися за принципом стислого опису змін: що було додано, змінено або видалено. Такий підхід забезпечував прозорість історії розробки та спрощував навігацію в репозиторії.

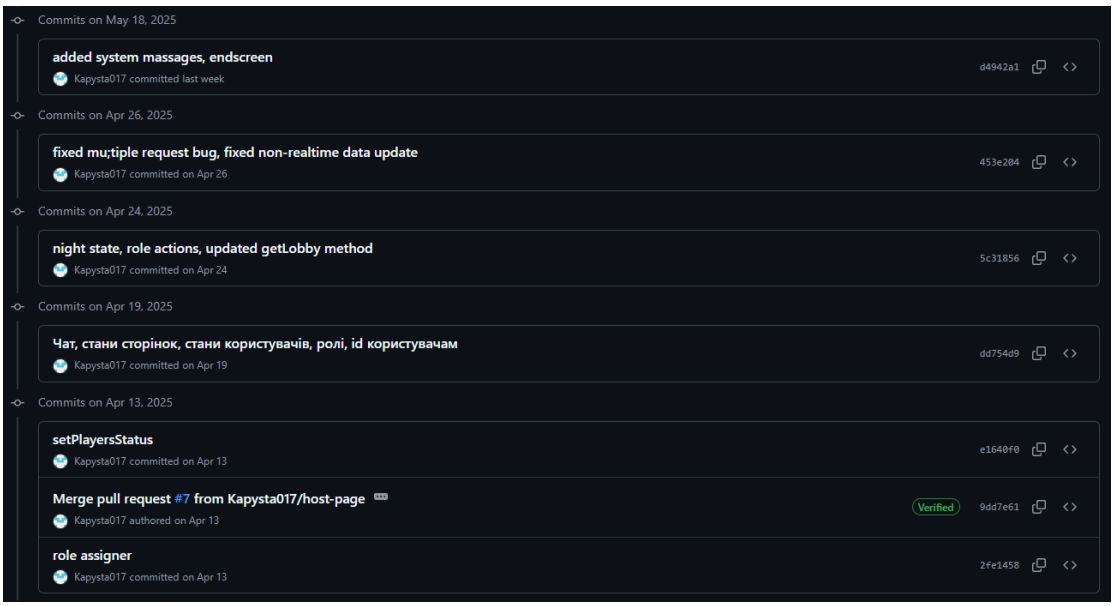

Рисунок 2.18 – Історія комітів

Файл README.md виконує роль вступної документації до проєкту. Він містить перелік основних технологій, що були використані під час розробки, а також інструкцію щодо локального запуску веб-застосунку.

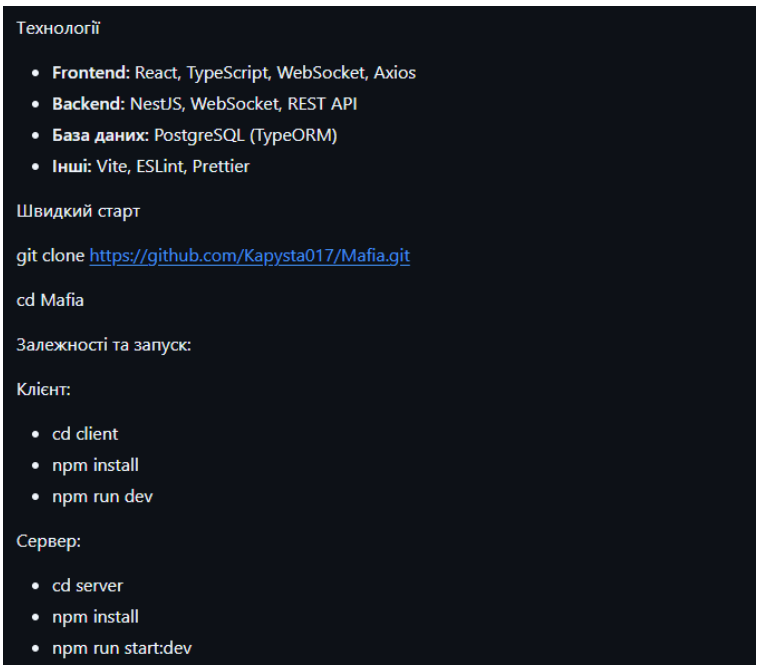

Рисунок 2.19 – Вигляд файлу README

У межах розробки проєкту система керування версіями Git та платформа GitHub значно допомогли в організації процесу роботи. Усі зміни в коді

фіксувались за допомогою комітів, що створювались безпосередньо через інтегровані інструменти Visual Studio Code. Назви комітів формувались згідно з принципом: стисло і зрозуміло описати суть змін, таких як додавання, модифікація чи видалення функціональності.

GitHub також забезпечував збереження історії змін, зручну навігацію по версіях та можливість швидко повертатись до попередніх станів проєкту в разі потреби. Це значно спрощувало процес тестування, верифікації вимог і командної співпраці.

2.4 Тестування та верифікація вимог

Для перевірки працездатності системи та її відповідності вимогам, було проведено низку тестів, які охоплювали основні сценарії використання. Верифікація функціоналу здійснювалась шляхом ручного тестування всіх доступних ролей, взаємодії між клієнтами, обробки типових і нетипових ситуацій у грі.

Таблиця 2.6 – Авторизація та реєстрація

Назва вимоги	Чи реалізовано?	Опис
Створення облікового запису з нікнеймом та паролем	Ні	Немає інструментів для створення облікового запису
Вхід до системи з подальшим збереженням сесії	Ні	Неможливість виконання в наслідок відсутності облікового запису
Можливість приєднання до гри без реєстрації як гість	Так	Основний метод взаємодії із системою

У результаті тестування функціональності авторизації та реєстрації було встановлено, що на поточному етапі проєкт не передбачає підтримки системи облікових записів користувачів. Відповідно, функціонал, пов’язаний із реєстрацією, входом та персоніфікованим збереженням даних, не реалізовано.

Таблиця 2.7 – Створення та приєднання до лобі

Назва вимоги	Чи реалізовано?	Опис
Створення нового ігрового лобі з унікальним ідентифікатором.	Так	Ідентифікатор лобі генерується автоматично при створенні кімнати
Приєднання до наявного лобі за кодом.	Так	На сторінці “Почати гру” є відповідне поле для введення коду.
Можливість обмежити доступ паролем або кількістю гравців.	Частково	Реалізовано обмеження по кількістю гравців проте за паролем ні
Наявність списку активних ігор до який можна приєднатись	Ні	Сторінка із списком активних лобі повністю не функціонуюча

За результатами тестування функціональності створення та приєднання до кімнат можна зробити висновок, що базовий механізм створення лобі та підключення за унікальним кодом реалізований і працює коректно. Водночас функціонал із захисту кімнат паролем та можливість здійснювати пошук лобі у списку очікування було відтерміновано на подальші етапи розробки або майбутні релізи. Через відсутність механізму передачі ролі адміністратора (ведучого) у разі його виходу з кімнати, лобі стає нефункціональним, що змушує гравців створювати нову кімнату. Крім того, певні налаштування, такі як кількість гравців, вибрані ролі та режим ведучого (ШІ чи ручний) не можуть бути змінені після створення кімнати. Це також обмежує гнучкість користувача й вимагає повторного створення лобі для внесення змін.

Таблиця 2.8 Керування грою

Назва вимоги	Чи реалізовано?	Опис
Встановлення кількості гравців та ролей у грі	Так	Кількість гравців задається при створенні кімнати
Таймер для обмеження часу на хід, обговорення чи голосування	Ні	В UI відповідні інструменти створенні, проте логіка не прописана
Можливість обирати між ШІ ведучим та ручним	Так	Обирається при створенні кімнати
Можливість задати ролі гравцям ручним та автоматичним методом	Частково	Реалізовано кнопки: “Задати ролі” та “Скинути ролі”
Можливість передати роль адміністратора гри іншому гравцю	Ні	Ніяк не реалізовано

Таблиця 2.9 – Геймплейна логіка

Назва вимоги	Чи реалізовано?	Опис
Автоматична зміна станів лобі в ігровому процесі	Так	Система автоматично змінює стани гри та повідомляє про це у чаті
Фаза ночі: можливість виконання ролями своїх дій	Частково	Реалізовані ролі: Мафія, Дон, Лікар
Фаза дня: обговорення, голосування, оголошення результатів	Так	Обговорення проходить у чаті, голосування реалізувано за допомогою контекстної кнопки, оголошення від системи реалізовані через повідомлення у чаті
Визначення переможців за правилами гри.	Так	Для реалізованих ролей передбачено та реалізовано два варіанти завершення гри: Перемога Мирних та Перемога Мафії

Геймплейна логіка реалізована майже повністю, за винятком повної підтримки різноманітних ролей. У майбутніх оновленнях доцільно розширити перелік доступних ролей та враховувати їхню специфіку в логіці підрахунку результатів гри.

Таблиця 2.10 – Чат між гравцями

Назва вимоги	Чи реалізовано?	Опис
Загальний чат у лобі	Так	Чат доступний від моменту створення кімнати до екрану завершення гри
Обмежений чат під час гри	Так	Чат блокується при певних станах гравця та під час ночі
Системні повідомлення про хід гри	Так	Система повідомляє користувачам усе через чат: Повідомлення про невдалі запити, зміна станів гри, результати ночі та голосування

В результаті тестування було встановлено, що чатова система працює коректно: повідомлення передаються в режимі реального часу, обробка різних типів чатів здійснюється відповідно до вимог, а інтерфейс користувача відображає повідомлення своєчасно та без збоїв. Таким чином, функціональність чату повністю відповідає заявленим вимогам на цьому етапі розробки.

У підсумку тестування можна зробити висновок, що хоча система наразі не реалізовує повного спектра функціональних вимог, ключові можливості, необхідні для проведення ігор, вже впроваджені та працюють стабільно. Це свідчить про те, що навіть на етапі початкових релізів застосунок має достатній рівень функціональності та включає значну кількість додаткових можливостей, які забезпечують зручність користування та позитивний користувацький досвід.

2.5 Результати розробки

На етапі проєктування було визначено архітектуру системи, розроблено логічну та фізичну структуру веб-сайту, спроєктовано модель і базу даних, а також побудовано бізнес-модель, яка описує основні взаємодії користувачів із системою. Особливу увагу приділено формуванню архітектури, що забезпечує масштабованість, модульність та зручність підтримки.

На етапі конструювання реалізовано ключові модулі серверної частини із застосуванням сучасного бекенд-фреймворку, а також розроблено основні компоненти клієнтської частини з інтерактивним графічним інтерфейсом користувача (GUI), що забезпечує зручну взаємодію користувача з системою.
Було також реалізовано повноцінну роботу з системою контролю версій Git, що дозволило організувати роботу, відслідковувати зміни в коді та підтримувати стабільність програмного комплексу.

У завершальному підрозділі проведено тестування та верифікацію функціональних вимог, що дозволило впевнитися у відповідності реалізованого програмного забезпечення початковим специфікаціям, перевірити правильність функціонування основних сценаріїв гри та забезпечити якість продукту.

Загалом, виконані роботи сформували повноцінний фундамент для функціонування веб-застосунку, що поєднує сучасні принципи проєктування, ефективну реалізацію й належний рівень перевірки якості.

РОЗДІЛ 3. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ТА ОХОРОНИ ПРАЦІ

3.1 Соціальні та психологічні фактори ризику в умовах ІТ-середовища

У сучасних умовах інтенсивного розвитку інформаційних технологій зростає важливість психосоціальних аспектів охорони праці. Професійна діяльність у сфері розробки програмного забезпечення часто супроводжується високими когнітивними навантаженнями, емоційним вигоранням, соціальною ізоляцією та порушенням режиму праці.

До основних соціальних і психологічних факторів ризику в ІТ-галузі належать:

- Хронічне перенавантаження через нестабільні графіки та овертайми.
- Монотонність праці при роботі з великими обсягами коду.
- Недостатня фізична активність, що спричиняє погіршення загального стану здоров’я.
- Високий рівень стресу, пов’язаний із дедлайнами, багами, відповідальністю за безперервну роботу систем.

Згідно з дослідженнями [21, с. 112], саме психоемоційне виснаження є однією з головних причин зниження продуктивності та підвищення кількості помилок у програмному коді. Психологічні ризики можуть також провокувати конфлікти в команді, зниження мотивації, а в довгостроковій перспективі — розвиток психосоматичних захворювань.

Одним із шляхів зниження таких ризиків є впровадження ергономічної організації праці, періодичне чергування розумової та фізичної активності, дотримання гнучкого, але збалансованого графіку роботи. Згідно з нормами [22] , кожні 2 години роботи з дисплейними пристроями має проводитися 15-хвилинна перерва, а для працівників віком до 18 років — не рідше ніж кожна година.

Психосоціальні небезпеки – це аспекти роботи, які можуть завдати психологічної або фізичної шкоди. Вони часто виникають через структуру роботи, робоче середовище та організаційні фактори . Це може включати високі

вимоги до роботи, низьку підтримку або вплив травматичних подій. Сам по собі стрес не є травмою. Але якщо працівники часто перебувають у стресі протягом тривалого часу або рівень стресу високий, це може завдати шкоди [23].

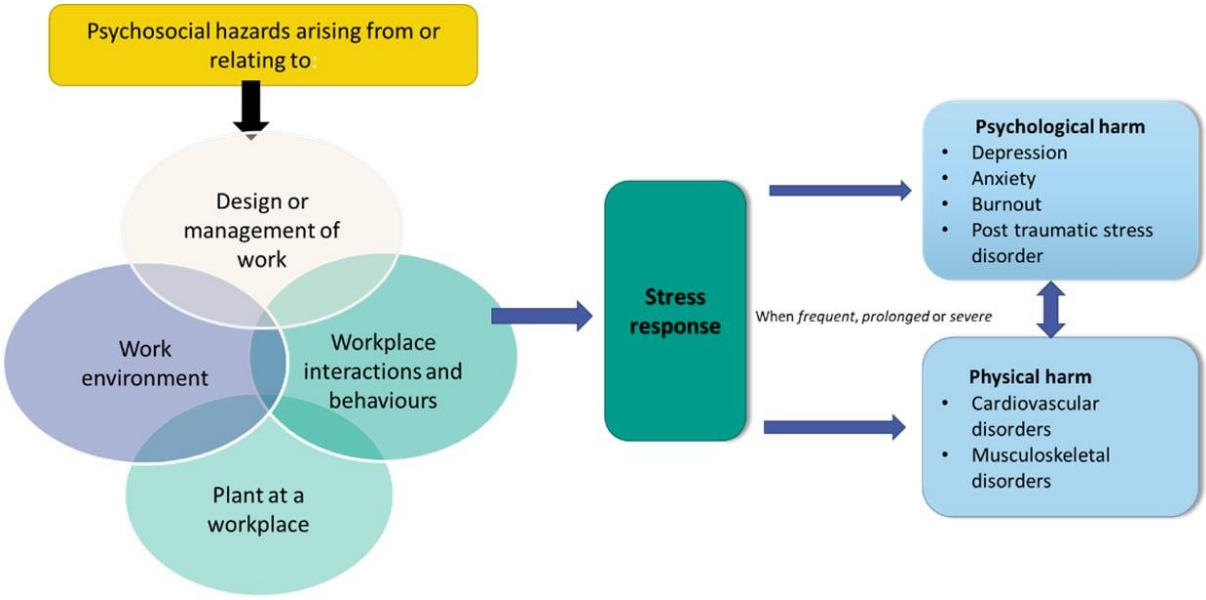

Рисунок 1 – Модель стресу, пов'язаного з роботою [23]

Крім того, створення психологічно комфортного мікроклімату в колективі, впровадження практик корпоративного ментального здоров’я, можливість звернення до фахівців з психічного здоров’я також вважається ефективним заходом [24, с. 140].

3.2 Заходи щодо захисту обладнання від короткого замикання

Комп’ютерне обладнання, сервери, маршрутизатори та інша апаратура є важливою складовою інфраструктури сучасних інформаційних систем. Однією з основних загроз їх надійній роботі є коротке замикання (КЗ), яке може спричинити не тільки пошкодження пристроїв, а й пожежу.

Коротке замикання виникає внаслідок порушення ізоляції провідників, помилок у монтажі або дії зовнішніх факторів (механічні пошкодження,

перевантаження, перенапруга тощо). Тому заходи захисту від КЗ повинні охоплювати організаційні, технічні та профілактичні рішення [25].

Серед технічних заходів:

- Встановлення автоматичних вимикачів та ПЗВ (пристроїв захисного вимкнення) відповідно до потужності споживачів [26].
- Використання багатоканальних стабілізаторів напруги та УЗІП (пристроїв захисту від імпульсних перенапруг).
- Регулярна перевірка стану кабельної ізоляції та контактних з’єднань (згідно з нормами [27]).

Крім того, рекомендовано застосовувати сертифіковані блоки живлення із захистом по струму та напрузі, що відповідають нормам [28]. Особливу увагу слід приділяти заземленню обладнання — воно повинно відповідати вимогам ПУЕ (Правил улаштування електроустановок) [29, розд. 1.7] .

Метою захисного заземлення є усунення небезпеки ураження людини електричним струмом при появі напруги на корпусі або на інших не струмоведучих металевих частинах ЕУ, тобто при замиканні на корпус (наприклад, при пробої ізоляції).

Дія захисного заземлення полягає у зменшенні до безпечної величини сили струму, що проходить через тіло людини при її дотику до корпусу ЕУ, що виявився під напругою. Це досягається зменшенням потенціалу корпусу заземленого устаткування [30].

Організаційні заходи включають:

- Інструктаж працівників щодо дій при виявленні перегріву або запаху горілого.
- Заборону використання саморобних подовжувачів і несертифікованих зарядних пристроїв.
- Систематичний технічний огляд електромереж і обладнання з боку кваліфікованих спеціалістів.

За дотримання вказаних норм і рекомендацій рівень електробезпеки значно підвищується, що мінімізує ризик аварій і простоїв у роботі систем [31, с. 173].

ВИСНОВКИ

Розробка веб-застосунку для гри "Мафія" пройшла кілька послідовних етапів, кожен із яких зробив свій внесок у формування цілісного й функціонального продукту.

На початковому етапі було проведено аналіз предметної області. Основна увага зосереджувалася на адаптації класичних правил гри "Мафія" до формату онлайн-взаємодії. Було визначено, які ролі та фази гри необхідно підтримувати, як має відбуватись комунікація між гравцями, та яким чином можна реалізувати функціонал автоматизованого ведучого.

Далі відбулося проектування архітектури системи. Було розроблено структуру бази даних і визначено основні сутності, необхідні для підтримки гри: лобі, гравці, налаштування, поточний стан гри, ролі тощо. Паралельно було спроектовано REST API та WebSocket-взаємодію між клієнтською та серверною частинами, що забезпечило як стабільну передачу даних, так і режим реального часу для синхронізації гри між учасниками.

На етапі конструювання відбулася реалізація серверної частини з використанням NestJS та клієнтської частини на базі React. Інтерфейс користувача розроблявся відповідно до попереднього дизайну, створеного у Figma, з урахуванням зручності взаємодії для нових і досвідчених користувачів. Було реалізовано ключові компоненти: створення та приєднання до лобі, управління налаштуваннями гри, чат, система фаз гри (ніч, день, голосування) та фінальний екран із результатами.

Завершальним етапом стало тестування та верифікація вимог. У ході тестування було підтверджено коректність основного функціоналу, зокрема створення лобі, підключення гравців, передача повідомлень у чаті, зміна фаз гри та завершення ігрового сеансу. Водночас виявлено деякі функціональні обмеження, такі як відсутність передачі ролі ведучого у разі його виходу, неможливість редагування налаштувань лобі після створення, а також

незавершеність реалізації деяких ролей, що впливає на точність визначення переможців.

Таким чином, у результаті розробки було реалізовано повноцінний прототип веб-застосунку, що дозволяє проводити ігри в "Мафію" в онлайн-режимі. Хоча частина запланованого функціоналу ще потребує доопрацювання, основна ідея та ключові механіки гри вже успішно реалізовані.

ВИКОРИСТАННІ ДЖЕРЕЛА

1. Verpex. 10 Most Popular Types of Websites [Електронний ресурс] // *Verpex Blog*. – 2023. – Режим доступу: https://verpex.com/blog/marketing-tips/10-most-popular-types-of-websites

2. Robinson B. Mafia: The Game. URL: https://mafiathegame.benrobinson.dev

3. Mafia.gg. Онлайн-версія гри Mafia URL: https://mafia.gg

4. Корлиханов О., Мудрик І. Розробка веб-сайту для онлайн гри «Мафія» з використанням технологій Node.js та фреймворку – React // VIII Міжнародна студентська науково-технічна конференція "Природничі та гуманітарні науки. Актуальні питання": зб. тез доповідей, 23–25 квітня 2025 р. – Тернопіль: ТНТУ ім. І. Пулюя, 2025. – С. 168.

5. Joshi, R. Best Practices for Combining ReactJS with NodeJS for Modern Web Applications [Електронний ресурс] / Rishi Joshi // *LinkedIn Pulse*. – 2023. – Режим доступу: https://www.linkedin.com/pulse/best-practices-combining-reactjs-nodejs-modern-web-applications-4yswf

6. What are Hybrid Development Methods Made Of? An Evidence-based Characterization. arXiv.org. URL: https://arxiv.org/abs/2101.08016

7. Sugiyanto Sugiyanto. The Development and Evaluation of Web-based Multiplayer Games with Imperfect Information using WebSocket URL: https://www.researchgate.net/publication/336162234_The_Development_and_Evaluation_of_Web-based_Multiplayer_Games_with_Imperfect_Information_using_WebSocket

8. Batri A. Jakobs Law for UX design. Medium. URL: https://arpit-batri.medium.com/jakobs-law-in-ux-design-dc88554a024b

9. What is fitts' law?. The Interaction Design Foundation. URL: https://www.interaction-design.org/literature/topics/fitts-law

10. The laws of proximity and common region in UX design | software country. Software Development Company | Software Country. URL: https://softwarecountry.com/company/our-blog/laws-of-proximity-in-ui/

11. Contentful. Using React with Node.js: A web development dream team [Електронний ресурс] // *Contentful Blog*. – 2022. – Режим доступу: https://www.contentful.com/blog/react-node-js

12. COI.ua. Website Structures: How to Choose the Best Option for Your Web Project [Електронний ресурс] // *COI Blog*. – 2023. – Режим доступу: https://coi.ua/blog/Cbc/Website-Structures-How-to-Choose-the-Best-Option-for-Your-Web-Project

13. https://www.researchgate.net/publication/374509386_Mastering_database_normalization_A_comprehensive_exploration_of_normal_forms

14. Yatsyshyn V., Pastukh O., Palamar A., Zharovskyi R. (2023) Technology of relational database management systems performance evaluation during computer systems design. Scientific Journal of TNTU (Tern.), vol. 109, no 1, pp. 54-65. URL: http://elartu.tntu.edu.ua/handle/lib/42076

15. Janicsák Ádám. Offline versus online multiplayer gaming experience: A comparison study about social deduction games [Електронний ресурс] / Á. Janicsák. – University of Twente, Master's thesis, 2024. – 58 с. – Режим доступу: https://purl.utwente.nl/essays/98177

16. Miu A., Ferreira F., Yoshida N., Zhou F. Generating Interactive WebSocket Applications in TypeScript [Електронний ресурс] // arXiv preprint. – 2020. – Режим доступу: https://arxiv.org/abs/2004.01321

17. Мартін Р. Чистий код: створення, аналіз і рефакторинг / пер. з англ. О. Карпенко. – Київ: Видавнича група «КМ-Букс», 2019. – 464 с.

18. Riven.ch. Build lobby-based online multiplayer browser games with React and Node.js [Електронний ресурс] // *Riven Blog*. – 2023. – Режим доступу: https://riven.ch/en/blog/build-lobby-based-online-multiplayer-browser-games-with-react-and-nodejs

19. Ably. The complete guide to WebSockets with React [Електронний ресурс] // *Ably Blog*. – 2023. – Режим доступу: https://ably.com/blog/websockets-react-tutorial

20. Kapysta017. Mafia: веб-реалізація гри «Мафія» / Kapysta017 [Електронний ресурс] // GitHub. – 2024. – Режим доступу: https://github.com/Kapysta017/Mafia

21. Охорона праці : навч. посіб. / Н.І. Андрейчук, Ю.В. Кіт, С.В. Шибанов, О.В. Шерстньова. Львів : Львів. політехніка, 2021. 276 с.

22. ДСанПін 3.3.2.007-98. Державні санітарні правила і норми роботи з відеодисплейними терміналами. 1998. Київ. | URL: https://zakon.rada.gov.ua/rada/show/v0007282-98#Text

23. Healthy Workplaces. Psychosocial hazards. URL: https://www.healthyworkplaces.sa.gov.au/action-areas/psychosocial-hazards

24. Бедрій Я.І. Основи охорони праці: навч. посібник. Тернопіль: Навчальна книга – Богдан, 2018. 240 с.

25. Коротке замикання: як запобігти пожежі. НМЦ ДСНС України. URL: https://nmc.dsns.gov.ua/zk/news/ostanni-novini/korotke-zamikannia-iak-zapobigti-pozezi.

26. НПАОП 0.00-2.24-05. Перелік робіт з підвищеною небезпекою. Затв. наказом Держгірпромнагляду № 15 від 26.01.2005. URL: https://zakon.rada.gov.ua/laws/show/z0232-05#Text

27. Про затвердження Гігієнічних вимог до умов праці при роботі з екранними пристроями. Затв. наказом МОЗ України № 207 від 14.02.2018. URL: https://zakon.rada.gov.ua/laws/show/z0508-18#Text

28. Про охорону праці: Закон України від 21.11.2002 р. № 229-IV. – Відомості Верховної Ради України. 2003. – № 2 – Ст. 10.

29. Правила улаштування електроустановок офіційне видання. - К.: Міненерговугілля України, 2017. - 617 с. - URL: http://elib.chdtu.edu.ua/e-books/4178

30. Електронне навчання ТНТУ. Лекція 17.5: Методи захисту в електроустановках. Курс: «Безпека життєдіяльності та охорона праці». 2025. – URL: https://dl.tntu.edu.ua/content.php?cid=289205

31. Жидецький В.Ц. Охорона праці користувачів комп'ютерів. Львів: Афіша, 2020. 176 с.

32. Методичні вказівки до виконання дипломної роботи освітнього рівня - бакалавр студентами усіх форм навчання для напряму підготовки 121 – Інженерія програмного забезпечення/ Укладачі : Петрик М.Р.,Михалик Д.М., Кінах Я.І., Гладьо С.В., Цуприк Г.Б. – Тернопіль : Вид-во ТНТУ імені Івана Пулюя, 2016 – 28 с.

33. О. Голинська, І. Мудрик. Роль CRM-системи у сучасних бізнес-процесах. Матеріали X науково-технічної конференції „Інформаційні моделі, системи та технології "Тернопільського національного технічного університету імені Івана Пулюя, ТНТУ, 2022. С.113.

ДОДАТКИ

ДОДАТОК А

Лістинги

Лістинг А.1 – метод createLobby

```
  createLobby(
    hostName: string,
    avatarId: number,
    playersNumber: number,
    mafiaNumber: number,
    roles: Roles[],
    aiAnswer: boolean,
  ) {
    const lobbyId = Math.random().toString(36).substring(2, 8);
    const hostPlayer: Player = { id: 1, username: hostName, avatarId
};

    const newLobby = {
      host: hostPlayer,
      players: aiAnswer ? [hostPlayer] : [],
      settings: { playersNumber, mafiaNumber, roles },
      state: { currentState: 'waiting' as PosibleStates },
      aiAnswer,
    };
    this.lobbies.set(lobbyId, newLobby);

    return lobbyId;
  }
```

Лістинг А.2 – методи реалізації нічної фази

```
 waitForPlayerAction(lobby: Lobby, roleName: string): Promise<void>
{
    return new Promise((resolve) => {
      const interval = setInterval(() => {
        const player = lobby.players.find(
          (p) => p.role === roleName && p.action === true,
        );
        if (player) {
          clearInterval(interval);
          player.action = false;
          resolve();
        }
      }, 500);
    });
  }
```

```
  async performNightPhase(lobbyId: string) {
    const lobby = this.lobbies.get(lobbyId);
    if (!lobby) return { message: 'Лобі не знайдено' };
    const activeRoles = lobby.settings.roles.filter(
      (role) => role.action === true && role.status === true,
    );

    for (const role of activeRoles) {
      lobby.state.activeRole = role.roleName;
      this.lobbyGateway.emitFullLobbyState(lobbyId);

      await this.waitForPlayerAction(lobby, role.roleName);
      lobby.state.activeRole = 'Ніхто';
    }

    if (lobby.nightActions?.mafiaTarget) {
      lobby.nightActions.mafiaTarget.alive = false;
    }
    if (lobby.nightActions?.healedPlayer) {
      lobby.nightActions.healedPlayer.alive = true;
    }
    this.checkWinCondition(lobbyId);
    if (lobby.state.currentState !== 'ended') {
      lobby.state.currentState = 'day';
      this.lobbyGateway.emitFullLobbyState(lobbyId);
    }
    this.lobbyGateway.server.to(lobbyId).emit('chatMessage', {
      username: 'Система',
      text: 'Ніч завершено!',
    });
    if (lobby.nightActions?.mafiaTarget) {
      if (lobby.nightActions.mafiaTarget.alive === false) {
        this.lobbyGateway.server.to(lobbyId).emit('chatMessage', {
          username: 'Система',
          text: `Гравця ${lobby.nightActions.mafiaTarget.username}
знайшли мертвим`,
        });
      }
    }
    return { message: 'Ніч завершено' };
  }

  performRoleAction(lobbyId: string, playerId: number, targetId:
number) {
    const lobby = this.lobbies.get(lobbyId);
    if (!lobby) return { message: 'Лобі не знайдено' };

    const targetName = lobby.players.find((p) => p.id ===
targetId)?.username;
    const player = lobby.players.find((p) => p.id === playerId);
    if (!player || !player.alive)
```

```
      return { message: 'Гравця не знайдено або він мертвий' };

    switch (player.role) {
      case 'Дон':
      case 'Мафія':
        this.roleActionService.performMafiaAction(
          lobby,
          playerId,
          targetId,
          targetName,
        );
        break;
      case 'Доктор':
        this.roleActionService.performDoctorAction(
          lobby,
          playerId,
          targetId,
          targetName,
        );
        break;
      case 'Коханка':
        this.roleActionService.performMistressAction(
          lobby,
          playerId,
          targetId,
          targetName,
        );
        break;
      default:
        return { message: 'У ролі немає активної дії' };
    }

    player.action = true;
  }
```

Лістинг А.3 – конфігурація маршрутизації

```
    <Router>
      <Layout>
        <Routes>
          <Route path="/" element={<HomePage />} />
          <Route path="/start" element={<StartGamePage />} />
          <Route path="/create" element={<CreateGamePage />} />
          <Route path="/list" element={<BrowsePage />} />
          <Route path="/profile" element={<ProfilePage />} />
          <Route path="/lobby/:lobbyId" element={<LobbyAwainting />}
/>
        </Routes>
      </Layout>
    </Router>
```

Лістинг А.4 – компонент ChatModal:

```
export function ChatModal({ avaibleChat, lobbyId, username }) {
  const [messages, setMessages] = useState([]);
  const [newMessage, setNewMessage] = useState("");
  const [chatOpen, setChatOpen] = useState(false);

  useEffect(() => {
    socket.emit("joinChat", lobbyId);
    socket.on("chatMessage", (message) => {
      setMessages((prev) => [...prev, message]);
    });

    return () => {
      socket.off("chatMessage");
    };
  }, [lobbyId]);

  const sendMessage = () => {
    if (!newMessage.trim()) return;
    socket.emit("chatMessage", { lobbyId, username, text:
newMessage });
    setNewMessage("");
  };

  return (
    <div className="chat_container">
      * інтерфейс чату *
    </div>
  );
}
```

Лістинг А.5 – Компонент EndScreen:

```
import { avatars } from "../../utils/avatars";
export function EndScreen({ settings, state, users }) {
  const usedRoles = settings.roles.map((roleObj) =>
roleObj.roleName);
  const peacefulPlayers = users.filter((u) =>
!usedRoles.includes(u.role));
  const getAvatarUrl = (avatarId) => {
    const avatar = avatars.find((avatar) => avatar.id === avatarId);
    return avatar ? avatar.url : "/avatars/default.png";
  };
  return (
    <div className="end_page_container">
      <h2>Переможці: {state.winner}</h2>
      <div className="end_page_grid">
        {settings.roles.map((roleObj) => {
          const roleName = roleObj.roleName;
```

```
          const playersWithRole = users.filter((u) => u.role ===
roleName);

          return (
            <div key={roleName} className="role_container">
              <div className="role_info">{roleName}:</div>
              <div className="players_list">
                {playersWithRole.map((player) => (
                  <div key={player.id} className="player_avatar">
                    <img
                      src={getAvatarUrl(player.avatarId)}
                      className="avatar"
                    />
                    <div className="hf">{player.username}</div>
                  </div>
                ))}
              </div>
            </div>
          );
        })}

        {peacefulPlayers.length > 0 && (
          <div className="role_container">
            <div className="role_info">Громадяни:</div>
            <div className="players_list">
              {peacefulPlayers.map((player) => (
                <div key={player.id} className="player_avatar">
                  <img src={getAvatarUrl(player.avatarId)}
classNamе="avatar" />
                  <div className="hf">{player.username}</div>
                </div>
              ))}
            </div>
          </div>
        )}
      </div>
    </div>
  );
}
```

ДОДАТОК Б

Тези конференції

Міністерство освіти і науки України
Тернопільський національний технічний університет
імені Івана Пулюя
Маріборський університет (Словенія)
Технічний університет в Кошице (Словаччина)
Каунаський технологічний університет (Литва)
Львівський національний університет
імені Івана Франка,
Гірничо-металургійна академія ім. Станіслава Сташиця (Польща)
Луцький національний технічний університет,
Чернівецький національний університет
імені Юрія Федьковича,
Вроцлавський економічний університет (Польща)
Університет технологій та економіки
імені Хелени Ходковської (Польща)
Донбаська державна машинобудівна академія

Студентське наукове товариство

VIII МІЖНАРОДНА

студентська науково - технічна конференція

"ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ.

АКТУАЛЬНІ ПИТАННЯ"

24-25 квітня 2025 р.

(збірник тез конференції)

Тернопіль 2025

Єршов В. **ВИКОРИСТАННЯ NODE.JS ТА ELFSIGHT ДЛЯ РОЗРОБКИ ФУНКЦІОНАЛЬНОГО ІНТЕРНЕТ-МАГАЗИНУ**	**160**
Недошитко Л., Жук В. **ТЕРМОЯДЕРНИЙ СИНТЕЗ (ТОКАМАК ITER)**	**161**
Занчук Т. **РОЗРОБКА ІНФОРМАЦІЙНОЇ ПЛАТФОРМИ ДЛЯ МЕДІАПРОДУКТІВ З ВИКОРИСТАННЯМ СУЧАСНИХ ТЕХНОЛОГІЙ**	**163**
Недошитко Л., Зеньо Д. **ДНК-КОМП'ЮТЕР**	**165**
Ільчій Б. **АВТОМАТИЗОВАНА .NET-СИСТЕМА ДЛЯ ВЕДЕННЯ ЕЛЕКТРОННОЇ МЕДЧИНОЇ ДОКУМЕНТАЦІЇ ПАЦІЄНТІВ**	**166**
Колбасюк В. **РОЗРОБКА ТА ДООПРАЦЮВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ІНТЕРАКТИВНОГО ВИВЧЕННЯ ІСТОРІЇ УКРАЇНИ З ВИКОРИСТАННЯМ ПЛАТФОРМИ UNITY**	**167**
Корлиханов О., Мудрик І. **РОЗРОБКА ВЕБ-САЙТУ ДЛЯ ОНЛАЙН ГРИ «МАФІЯ» З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ NODE.JS ТА ФРЕЙМВОРКУ – REACT**	**168**
Кшивак Н. **РОЗРОБКА ВЕБ-ПЛАТФОРМИ ДЛЯ НАВЧАННЯ СЛІПОГО НАБОРУ ТЕКСТУ НА КЛАВІАТУРІ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ DJANGO**	**170**
Лунак О. **РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ДЛЯ ЗБОРУ ТА ОПРАЦЮВАННЯ ДАНИХ З ВИКОРИСТАННЯМ СУЧАСНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**	**171**
Макоїд О. **ЗАСТОСУВАННЯ ПРИНЦИПІВ LSB ТА SPLIT SPECTER ДЛЯ ЗАХИСТУ АУДІО ТА ШИФРУВАННЯ ДАНИХ ВСЕРЕДИНІ НИХ**	**172**
Марущак К. **ВИКОРИСТАННЯ ІНТЕЛЕКТУАЛЬНИХ АГЕНТІВ В IT-ОСВІТІ: МОЖЛИВОСТІ ТА ВИКЛИКИ**	**174**
Мацюк М. **АДАПТИВНИЙ ІНТЕРФЕЙС ВЕБСАЙТУ З УРАХУВАННЯМ ПРИНЦИПІВ UX / UI ДИЗАЙНУ**	**176**

VIII Міжнародна студентська науково - технічна конференція
"ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ. АКТУАЛЬНІ ПИТАННЯ"

УДК 004.4

Корлиханов О., Мудрик І.
Тернопільський національний технічний університет імені Івана Пулюя

РОЗРОБКА ВЕБ-САЙТУ ДЛЯ ОНЛАЙН ГРИ «МАФІЯ» З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ NODE.JS ТА ФРЕЙМВОРКУ – REACT

Korlykhanov O., Mudryk I.
Ternopil Ivan Puluj National Technical University

WEBSITE DEVELOPMENT FOR THE ONLINE GAME "MAFIA" USING NODE.JS TECHNOLOGIES AND THE FRAMEWORK – REACT

Наявність онлайн аналогів відомих настільних ігор не є рідкістю в сучасному світі. Проте в Українськомовному середовищі ринок не настільки розвинений. Розробка аналогів з використанням сучасних технологій позитивно вплине на позицію України у сфері розробки веб-застосунків, закривши недостатні для сучасного українського користувача потреби у веб-продуктах.

Для якісної реалізації проектів у даній ніші, варто застосовувати технології Node.js та фреймворк – React. Застосунки, розроблені з метою закрити потребу українських користувачів в даній сфері, з використанням більш доцільних та сучасних технологій, з підтримкою українською мови та покращеним візуальним інтерфейсом важливі для розвитку українського ринку.

Проаналізовавши основні типи архітектури веб-сайтів подібного типу, були обрані основні патерни та структури програмування, доцільні для даної сфери. Архітектура, заснована на деревоподібній структурі сайту, є доцільним вибором для специфікації розробки програмного забезпечення такого типу. [1] Основні принципи цієї структури полягають у розгалужені сторінок веб-сайту аналогічно гілкам від кореня дерева. Потреба у великій кількості унікальних сторінок, підтверджує доцільність цієї структури [2].

Щодо технологій, що базуються на Node.js, варто особливо виділити WebSocket та фреймворк Nest.js. Nest.js є сучасним інструментом для побудови масштабованих серверних застосунків, що базується на архітектурних принципах Angular і використовує TypeScript. Завдяки цьому фреймворку доцільно реалізовувати серверну логіку, дотримуючись чіткої структури модулів, контролерів та сервісів.

Такий підхід забезпечує високу читабельність коду, його легке тестування та повторне використання, що є важливим при розробці складних багатофункціональних систем. Це дозволить ефективно керувати ігровими процесами, зручно обробляти запити користувачів та швидко вносити зміни до логіки застосунку[3].

WebSocket, в свою чергу, є невід'ємною частиною таких проєктів, оскільки забезпечує двосторонній обмін повідомленнями в реальному часі між клієнтом і сервером. Це критично важливо для багатокористувацьких ігор, де затримка в передачі інформації може значно вплинути на ігровий процес та досвід користувача. Реалізація WebSocket дозволяє досягти миттєвої реакції системи на дії гравців, що значно покращує інтерактивність та динаміку гри.

Також, в подібних проектах необхідно застосовувати бази даних. Діалект SQL – PostgreSQL, є доцільним вибором, оскільки забезпечую швидкий обмін даними з сервером , що є критично важливим для оптимізації та комфортної взаємодії з

користувачем. Реляційна структура бази даних дозволяє ефективно організувати зв'язки між сутностями та забезпечити швидкий доступ до необхідної інформації. А розширені вбудовані типи даних забезпечують гнучкість розробки та організації взаємодії з іншими обраними технологіями[4].

Оскільки, проекти з розробки веб-ігор потребують часте оновлення багатьох компонентів, що може значно навантажувати сервер великою кількістю запитів. Тому для забезпечення оптимізації оновлення інформації на сторінках, використання фреймворку React необхідно.

Данна технологія дозволить створити динамічний, інтуїтивно зрозумілий інтерфейс з компонентною архітектурою, що ефективно вирішує вищеописану проблему. Також спрощує масштабування та усуває проблему повторного використання коду. React забезпечує ефективну роботу з DOM завдяки використанню віртуального DOM та підтримує швидке оновлення інтерфейсу при зміні стану. [5]

Отже, оптимальне технологічне рішення включає використання Node.js із Nest.js для побудови структурованої та масштабованої серверної логіки, WebSocket для забезпечення взаємодії в реальному часі, а також PostgreSQL як продуктивної реляційної бази даних. На клієнтській стороні доцільно використовувати React, що дозволяє створити динамічний інтерфейс, зручний у підтримці та масштабуванні.

Обрана деревоподібна архітектура сайту відповідає специфіці застосунків цього типу, де необхідна велика кількість унікальних сторінок та логічна організація контенту. Комплексне застосування зазначених технологій та архітектурних підходів дозволить створювати сучасні, стабільні та зручні веб-продукти, орієнтовані на українського користувача.

Список використаних джерел:

1. COI.UA. Website Structures: How to Choose the Best Option for Your Web Project URL: https://coi.ua/blog/Cbc/Website-Structures-How-to-Choose-the-Best-Option-for-Your-Web-Project/
2. Ivan Osiichuk, Vitaly Brevus, Dmytro Bishchak, Yaroslav Mashtaliar, Ivan Mudryk: Leveraging graphics tablet and JPen library to detect essential tremor. CITI 2024: 111-126
3. Turing. What is NestJS & Why Use It? URL: https://www.turing.com/blog/what-is-nest-js-why-use-it
4. Data Life UA. Postgres is Eating the Database World. URL: https://data-life-ua.com/db/postgres-is-eating-the-database-world/
5. Ranktracker. Why React JS is the Most Favored Front-End Technology for Startups URL: https://www.ranktracker.com/uk/blog/why-react-js-is-the-most-favored-front-end-technology-for-startups/

ДОДАТОК В

Діаграми

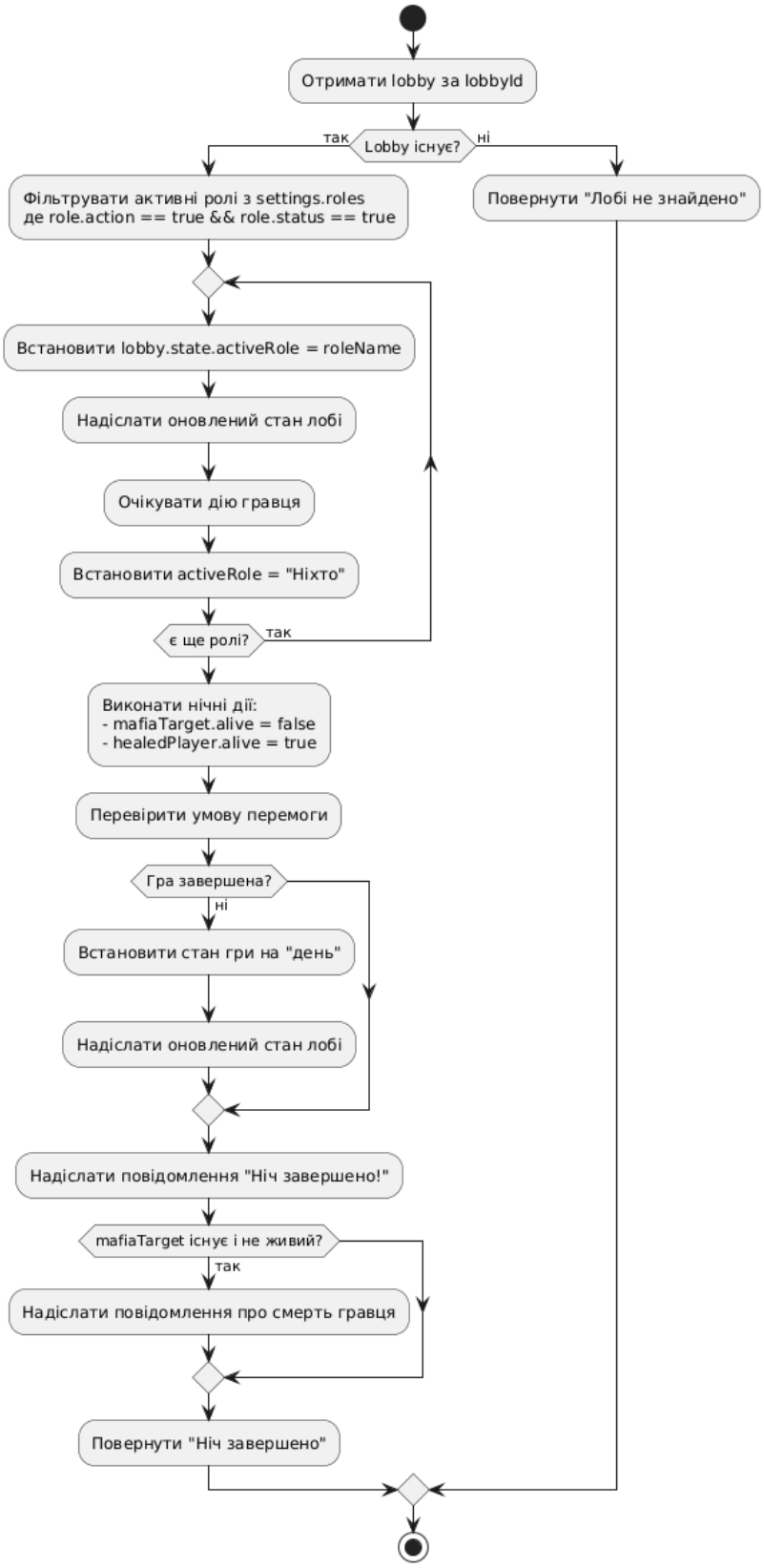

Рисунок В.1 – Блок-схема методів нічних дій

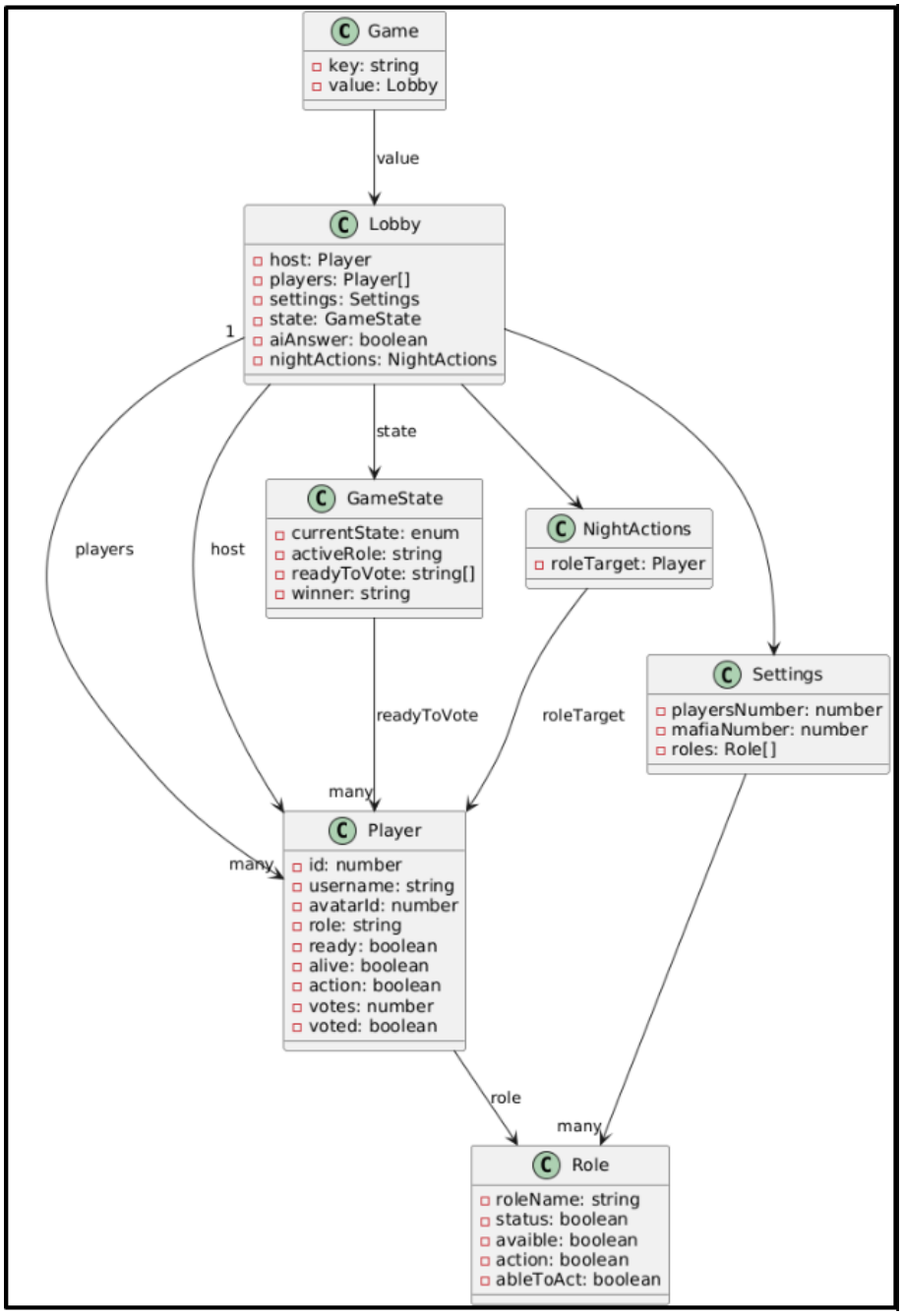

Рисунок В.2 – Діаграма сутностей

ДОДАТОК Г

Диск із проєктом