Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська дипломна робота на тему:

РОЗРОБКА WEB-ЗАСТОСУНКУ ДЛЯ ВЕДУЧОГО ГРИ "МАФІЯ"

Виконав: студент 4 курсу, групи 3КН-20б
спеціальності 122 Компютерні науки
Доценко Ілля Костянтинович
Керівник: к.т.н., професор каф.КН
Савчук Т.О.
«07» 06 2024 р.
Рецензент: Мізерний В.М.,
«18» 06 2024 р.

Допщщено до захисту
Зав. кафедри КН Яровий А.А.
«07» 06 2024 р.

Вінниця ВНТУ – 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань - 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри КН

проф., д.т.н. Яровий А.А.
"07" 03 2024р.

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТА

Доценку Іллі Костянтиновичу

1. Тема роботи: Розробка WEB-застосунку для ведучого гри «Мафія»
Керівник роботи: професор Савчук Тамара Олександрівна.
Затверджено наказом вищого навчального закладу "11" 03 2024 року № 80
2. Термін подання студентом роботи 27.05.2024
3. Вихідні дані до роботи:

- Кількість гравців - 10

- Кільість голосів – не меньше 1

- Кількість ролей мафія - 2

- Мова програмування - Об'єктно-орієнтована мова програмування

4. Зміст текстової частини(перелік питань, які потрібно зробити):
Вступ, Аналіз існуючих засобів гри "Мафія", , Формулювання мети роботи, предмета, об'єкта та задач подальшого дослідження; Висновки; Список використаних джерел.
5. Перелік ілюстративного матеріалу(з точним значенням обов'язкових креслень):

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-2	Савчук Т.О., професор		

7. Дата видачі завдання 07.03.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітк
		початок	Закінчення	
1	Аналіз існуючих засобів гри “Мафія”	06.05.2024	10.05.2024	
2	Аналіз сучасних засобів гри “Мафія”	11.05.2024	13.05.2024	
3	Застосування підходів розширення функціональності гри “Мафія”	14.05.2024	15.05.2024	
4	Постановка задачі	16.05.2024	17.05.2024	
5	Розробка удосконаленого аглоритму для ведучого гри “Мафія”	18.05.2024	22.05.2024	
6	Розробка функціонально удосконаленого узагальненого алгоритму гри “Мафія”	22.05.2024	24.05.2024	
7	Розробка алгоритму вбивства та лікування	24.05.2024	25.05.2024	
8	Розробка алгоритму додавання унікальних ролей	25.05.2024	26.05.2024	
9	Розробка структури web-застосунки для ведучого гри мафія	27.05.2024	29.05.2024	
10	Розробка інтерфейсу web-Розробка складових web-застосунку для ведучого гри “Мафія”	29.05.2024	31.05.2024	

13	Розроюка інтерфейсу web-застосунку	01.06.2024	03.06.2024	
14	Тестування функціонування web-застосунку	04.06.2024	05.06.2024	
15	Експериментальні дослідження	06.06.2024	06.06.2024	

Студент Доценко І.К.

Керівник роботи Савчук Т.О.

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 82 сторінок формату А4, на яких є 27 рисунків, список використаних джерел містить 20 найменувань.

Бакалаврська дипломна робота є частиною комплексної бакалаврської дипломної роботи, присвяченої створенню веб-застосунку гри “Мафія” для ведучого гри. В даній бакалаврській роботі розроблені форми та налаштування веб-застосунку гри “Мафія” для ведучого гри з використанням мов програмування JavaScript, каскадних таблиць стилів CSS та мови розмітки гіпертексту HTML.

Ключові слова: вебресурс, вебсайт, гра, форми, налаштування.

ABSTRACT

The bachelor thesis consists of 82 pages of A4 format, on which there are 27 figures, the list of used sources contains 20 names.

The bachelor's thesis is part of a comprehensive bachelor's thesis devoted to the creation of a web application for the game "Mafia" for the host of the game. In this bachelor's thesis, the forms and settings of the web application of the game "Mafia" for the host of the game were developed using JavaScript programming languages, CSS cascading style sheets and HTML hypertext markup language.

Keywords: web resource, website, game, forms, settings.

ЗМІСТ

ВСТУП

Актуальність досліджень. За часів карантину стало важко зустрітися великою компанією, щоб зіграти настільну гру. Через це почали розвиватися онлайн сервіси різноманітних ігор, спільного перегляду фільмів та інших подібних.

В сучасному цифровому світі веб-додатки стають все більш інтелектуальними та інтерактивними, відповідаючи на зростаючі вимоги користувачів. У зв'язку з цим розробка інтелектуальних веб-застосунків стає актуальною задачею, яка має великий потенціал у вирішенні різноманітних завдань та покращенні користувацького досвіду.

Гра "Мафія" є популярною інтерактивною грою, яка потребує стратегічного мислення та взаємодії між гравцями. Інтеграція цієї гри в онлайн-середовище вимагає розробки інтелектуального веб-застосунку, який буде не тільки відтворювати механіку гри, але й забезпечувати інтелектуальну систему управління та взаємодії з гравцями. Такий веб-застосунок відкриває нові можливості для онлайн-ігор, забезпечуючи більш широкий та захоплюючий досвід для гравців. Що і визначає актуальність дослідження задачі.

Метою дослідження є розширення функціональних можливостей для ведучого гри “Мафія”.

Об'єктом дослідження є процес гри “Мафія” для ведучого.

Предметом дослідження є програмні засоби для ведучого гри “Мафія”.

Задачі дослідження:

1. Аналіз існуючих засобів гри “Мафія”
2. Розробка удосконаленого алгоритму для ведучого гри “Мафія”
3. Розробка структури web-застосунку для ведучого гри “Мафія”
4. Розробка складових web-застосунку для ведучого гри “Мафія”
5. Експериментальні дослідження

Особистий внесок здобувача.

Усі наукові та прикладні результати, викладені у бакалаврській дипломній роботі, отримані автором особисто.

Технічне завдання наведено в додатку А.

Достовірність отриманих результатів засвідчується узгодженістю кінцевого практичного завдання із початковими теоретичними дослідженнями.

Практична цінність отриманих результатів

Розроблено web-застосунок для ведучого гри “Мафія”, який відрізняється підвищеними функціональними можливостями за рахунок додавання нових функцій та ролей.

1. Розроблено удосконалений алгоритм для ведучого гри “Мафія”
2. Розроблено алгоритм вбивста та лікування гравців
3. Розроблено алгоритм утворення унікальних ролей
4. Програмно реалізовано складові веб-застосуноку для ведучого гри мафія

1 АНАЛІЗ ІСНУЮЧИХ ЗАСОБІВ ГРИ “МАФІЯ”

1.1 Аналіз сучасних засобів гри “Мафія”

Серед засобів виділено найпооулярніші та з найкращими функціональними можливостями. А саме:

1) Mafia: The Game [1]

"Mafia: The Game" - це веб-платформа, яка пропонує онлайн-версію популярної гри "Мафія". На цьому сайті гравці з усього світу можуть зустрічатися та взаємодіяти, долучаючись до захоплюючих інтерактивних гральних сеансів. Дана платформа має такі можливості:

Онлайн гра в мафію, організація ігор з можливістю створення приватних та публічних лобі, автоматизовані процеси, такі як голосування та нічні дії, користувацький профіль, текстовий чат для спілкування під час гри, інструменти для модераторів та ведучих для управління ігровим процесом, мобільна версія та додатки, інтерфейс для управління голосуванням та нічними діями.

Ці функціональні можливості роблять "Mafia: The Game" комплексною платформою для шанувальників гри в мафію, забезпечуючи їх усім необхідним для організації, проведення та аналізу ігор, проте вона не має функціональних можливостей для створення унікальних ролей та маніпулювання таймером.

2) Polemica [2]

"Polemica" - це онлайн-платформа для гри в "Мафію", яка створена з метою надати гравцям можливість грати інтерактивну гру, взаємодіючи з іншими учасниками з усього світу. Дана платформа має такі можливості:

Організація дебатів, можливість створення публічних та приватних дебатів, користувацький профіль, текстовий чат для спілкування під час дебатів, форуми та обговорення для обміну ідеями, стратегіями та досвідом, система груп та клубів для організації тематичних об’єднань, мобільна версія та додатки:

Підтримка мобільної версії сайту для участі в дебатах зі смартфонів та планшетів. Ці функціональні можливості роблять Polemica потужною платформою для проведення онлайн-дебатів, надаючи користувачам усі необхідні інструменти для організації, участі та аналізу дебатів, проте не має новітніх функціональних можливостей, як утворення унікальних ролей та завчасного припинення голосування.

3) Mafia.GG [3]

MafiaGG — це онлайн-платформа для гри в мафію, яка надає користувачам різноманітні функціональні можливості для комфортної гри та взаємодії. Ось основні функціональні можливості сайту:

Організація ігор з можливістю створення приватних та публічних лобі, різні режими гри: класична мафія, кастомні режими, турнірні ігри, користувацький, система рейтингу та досягнень, що відображає прогрес і успіхи гравця, текстовий чат для спілкування під час гри, стрімінг ігор та можливість перегляду записів минулих ігор, організація турнірів, підтримка мобільної версії сайту для гри на смартфонах та планшетах.

Ці функціональні можливості роблять MafiaGG потужним інструментом для шанувальників гри в мафію, надаючи їм усі необхідні інструменти для організації, проведення та аналізу ігор, проте немає функціональних можливостей, як утворення унікальних ролей та маніпулювання таймером та голосуванням.

Таблиця 1.1 – Порівняння функціональних можливостей засобів.

Критерій	Mafia:The Game	Mafia.GG	Polemica
Можливість додавання унікальних ролей	-	-	+
Можливість запуску таймера	-	+	-
Можливість завчасного припинення голосування	+	-	-

Отже, серед розглянутих web-застосунків найкращим є Mafia:The Game з своїм комфортним та зрозумілим інтерфейсом, та розвиненими функціональними можливостями порівняно з іншими аналогами.

1.2 Застосування підходів розширення функціональності гри “Мафія”

Розширення функціональності сайтів гри "Мафія", може значно покращити користувацький досвід та зробити гру більш цікавою та інтерактивною. Ось кілька ключових аспектів, на які можна звернути увагу:

1. Покращення Інтерфейсу та Навігації

Інтуїтивно зрозумілий дизайн: Забезпечити простоту і легкість у використанні, щоб новачки швидко розуміли, як почати гру.

Адаптивний дизайн: Оптимізація сайту для мобільних пристроїв, планшетів і різних розмірів екранів.

2. Соціальні Функції

Чати та форуми: Інтеграція чату для обговорень гри в реальному часі та форуми для довготривалих дискусій і обміну стратегіями.

Система друзів: Можливість додавати інших гравців у друзі та створювати приватні ігрові кімнати.

3. Інтерактивні Елементи

Анімовані аватари: Можливість створювати та використовувати анімовані аватари для більшого залучення гравців.

Ігрові предмети та нагороди: Впровадження системи нагород і досягнень, які можна заробляти під час гри.

4. Поліпшення Геймплею

Різноманітні режими гри: Введення різних режимів гри, таких як класична "Мафія", "Шериф проти мафії" та інші варіанти.

Детальна статистика: Відстеження і публікація статистики гравців, таких як кількість виграних/програних ігор, успішних обманів тощо.

5. Технічні Покращення

Стабільність та продуктивність: Оптимізація коду для швидкої роботи сайту без затримок і багів.

Безпека: Захист даних користувачів та запобігання шахрайству.

6. Інтеграція з Соціальними Мережами

Авторизація через соцмережі: Можливість входу на сайт за допомогою облікових записів соціальних мереж.

Поділ результатів гри: Опція для гравців ділитися своїми результатами та досягненнями у соціальних мережах.

7. Освітні Матеріали

Гайди та відеоуроки: Створення розділу з гайдами, стратегіями та відеоуроками для нових гравців.

Форум Питань та Відповідей: Платформа для обміну знаннями та порадами між гравцями.

8. Монетизація

Преміум підписки: Введення платних підписок, які надають доступ до ексклюзивних ігрових режимів або особливих можливостей.

Внутрішньоігрові покупки: Продаж унікальних аватарів, скілів та інших ігрових предметів.

Розширення функціональності сайту для ведучого гри "Мафія" не тільки зробить його більш привабливим для поточних гравців, але й залучить нову аудиторію, забезпечуючи захоплюючий і інтерактивний ігровий досвід.

Оптимізація сайтів є ключовим чинником успішної присутності в Інтернеті, оскільки вона забезпечує підвищення швидкості завантаження сторінок, поліпшення користувацького досвіду та підвищення видимості в пошукових системах. У сучасному конкурентному середовищі якісна оптимізація дозволяє залучити більше відвідувачів, підвищити конверсію та забезпечити сталий ріст бізнесу в цифровій сфері.

1. Оптимізація швидкості завантаження[6]

Оптимізація швидкості завантаження веб-сайту включає методи та технології, що дозволяють зменшити час завантаження сторінок, покращуючи користувацький досвід. Швидкі сайти забезпечують вищу задоволеність користувачів, зменшення показника відмов, підвищення конверсій і кращі SEO-рейтинги. Це досягається через мінімізацію HTTP-запитів, стиснення файлів, впровадження кешування, оптимізацію зображень і використання контент-мереж доставки (CDN). У сучасному цифровому середовищі, де кожна секунда завантаження впливає на поведінку користувачів, оптимізація швидкості є критично важливим аспектом успішної онлайн-присутності.

Переваги:

Покращений користувацький досвід: Швидкі сайти приваблюють більше користувачів і зменшують показник відмов.

Підвищення SEO-рейтингів: Пошукові системи надають перевагу швидким сайтам.

Збільшення конверсій: Швидке завантаження сприяє підвищенню кількості конверсій.

Недоліки:

Складність реалізації: Вимагає технічних знань і досвіду.

Постійне оновлення: Необхідно регулярно підтримувати та оновлювати налаштування для збереження швидкості.

2. Оптимізація для пошукових систем (SEO)[7]

Оптимізація для пошукових систем (SEO) включає методи та стратегії, спрямовані на підвищення видимості веб-сайту у пошукових системах. Основні аспекти SEO включають on-page оптимізацію, яка охоплює створення якісного контенту, використання ключових слів, оптимізацію заголовків, метаописів та зображень. Off-page SEO зосереджується на отриманні зворотних посилань, соціальних сигналах та побудові авторитету сайту. Технічне SEO забезпечує правильну індексацію та зручність для пошукових роботів через оптимізацію

швидкості завантаження, структури URL, файлу robots.txt та карти сайту (sitemap). Ефективна SEO-стратегія сприяє збільшенню органічного трафіку, підвищенню позицій у пошукових результатах та довіри користувачів, що є критично важливим для сталого розвитку бізнесу в онлайн-середовищі.

Переваги:

Збільшення органічного трафіку: Поліпшення видимості сайту у пошукових системах.

Підвищення довіри: Сайти, що займають високі позиції, викликають більше довіри у користувачів.

Довгострокові результати: Ефективна SEO-стратегія приносить стабільний трафік на тривалий час.

Недоліки:

Тривалий процес: SEO-оптимізація потребує часу для досягнення видимих результатів.

Конкуренція: Висока конкуренція може ускладнити досягнення високих позицій.

3. Мобільна оптимізація[8]

Мобільна оптимізація включає адаптацію веб-сайту для забезпечення зручного та ефективного користування на мобільних пристроях. Основні аспекти мобільної оптимізації охоплюють використання адаптивного дизайну, що дозволяє сайту автоматично підлаштовуватися під різні розміри екранів, та впровадження прискорених мобільних сторінок (AMP), які зменшують час завантаження. Крім того, оптимізація контенту, зображень і навігації для мобільних користувачів значно покращує користувацький досвід. Мобільна оптимізація сприяє підвищенню позицій у пошукових системах, оскільки Google надає перевагу мобільно адаптованим сайтам. Вона також розширює аудиторію, підвищує задоволеність користувачів та сприяє збільшенню конверсій, що є важливим для успішної онлайн-присутності в сучасному мобільному світі.

Переваги:

Розширення аудиторії: Більше користувачів мають доступ до сайту через мобільні пристрої.

Покращення мобільного UX: Підвищення зручності використання сайту на мобільних пристроях.

SEO переваги: Google надає перевагу мобільно оптимізованим сайтам у пошукових результатах.

Недоліки:

Додаткові витрати: Потребує додаткових ресурсів і часу для реалізації.

Підтримка різних пристроїв: Необхідність врахування великої кількості різних мобільних пристроїв і їхніх характеристик.

4. Безпека сайту[9]

Безпека сайту охоплює заходи та практики, спрямовані на захист веб-сайту від зловмисних атак, втрати даних та несанкціонованого доступу. Основні аспекти безпеки включають впровадження SSL-сертифікатів для забезпечення захищеного з'єднання HTTPS, регулярне оновлення програмного забезпечення та плагінів, використання брандмауерів та захист від DDoS-атак. Крім того, безпека сайту передбачає створення резервних копій даних та використання складних паролів. Ефективні заходи безпеки підвищують довіру користувачів, захищають конфіденційні дані та запобігають фінансовим втратам. У сучасному цифровому середовищі, де кіберзагрози стають все більш поширеними, забезпечення безпеки сайту є критично важливим для стабільної та надійної онлайн-присутності.

Переваги:

Захист даних: Підвищення безпеки сайту захищає дані користувачів та запобігає зловмисним атакам.

Підвищення довіри: Безпечні сайти викликають більше довіри у користувачів.

Підвищення SEO: Пошукові системи віддають перевагу безпечним сайтам.

Недоліки:

Витрати: Впровадження та підтримка безпеки можуть бути дорогими.

Складність: Реалізація ефективних заходів безпеки потребує високих технічних знань.

5. Оптимізація користувацького досвіду (UX)[10]

Переваги:

Підвищення залученості: Зручний та інтуїтивний дизайн покращує взаємодію користувачів із сайтом.

Зменшення показника відмов: Поліпшення UX зменшує кількість користувачів, які покидають сайт одразу після заходу.

Позитивні відгуки: Задоволені користувачі частіше залишають позитивні відгуки та рекомендації.

Недоліки:

Висока вартість: Професійна UX-дизайнерська робота може бути дорогою.

Часові витрати: Проектування та тестування UX можуть займати багато часу.

Оптимізація користувацького досвіду (UX) полягає в створенні ефективного та задовільного взаємодії користувача з веб-сайтом. Це включає в себе розробку інтуїтивного і зручного інтерфейсу, оптимізацію швидкості завантаження сторінок, покращення навігації та організації контенту. Основні принципи UX допомагають зробити веб-сайт більш привабливим та легким у використанні для користувачів, що призводить до збільшення задоволеності, зменшення відмов та підвищення конверсій. Успішна оптимізація UX сприяє покращенню репутації бренду, підвищенню лояльності користувачів та стабільному розвитку веб-проекту.

6. Оптимізація контенту[11]

Переваги:

Підвищення залученості: Якісний та релевантний контент приваблює більше користувачів.

SEO-переваги: Оптимізований контент сприяє кращій індексації сайту пошуковими системами.

Підвищення авторитету: Високоякісний контент підвищує авторитет сайту та довіру користувачів.

Недоліки:

Трудомісткість: Створення якісного контенту потребує значних зусиль і часу.

Постійне оновлення: Необхідно регулярно оновлювати контент для підтримання його актуальності.

Оптимізація контенту - це процес створення та оптимізації вмісту веб-сайту з метою підвищення його привабливості для користувачів та пошукових систем. Цей процес включає в себе використання ключових слів, оптимізацію заголовків та мета-описів, створення якісного та цільового контенту для аудиторії, а також використання відео, зображень та інших мультимедійних форматів. Ефективна оптимізація контенту допомагає привернути більше трафіку на сайт, підвищити його позиції в пошукових результатах і покращити загальний користувацький досвід.

7. Аналітика та моніторинг[12]

Переваги:

Покращене розуміння поведінки користувачів: Аналітика дозволяє виявити сильні та слабкі сторони сайту.

Прийняття обґрунтованих рішень: Дані аналітики допомагають приймати обґрунтовані бізнес-рішення.

Оптимізація маркетингових стратегій: Моніторинг дозволяє краще налаштувати маркетингові кампанії.

Недоліки:

Вартість інструментів: Деякі аналітичні інструменти можуть бути дорогими.

Складність аналізу: Аналіз великих обсягів даних потребує спеціальних знань і навичок.

Аналітика та моніторинг включають збір, аналіз та використання даних для оцінки ефективності веб-сайту та маркетингових кампаній. Ці процеси дозволяють виявляти ключові тренди, розуміти поведінку користувачів, визначати найбільш успішні канали приваблення трафіку і вчасно реагувати на зміни. Застосування аналітики допомагає підвищити ефективність маркетингу, збільшити конверсії та досягнути більшої рентабельності інвестицій в цифровий маркетинг. Моніторинг дозволяє вчасно виявляти проблеми та вдосконалювати стратегії, щоб досягти більшого успіху в онлайн-середовищі.

Кожен тип оптимізації має свої переваги та недоліки, які потрібно враховувати при розробці та підтримці веб-сайту. Комплексний підхід до оптимізації дозволяє досягти найкращих результатів та забезпечити стабільний розвиток бізнесу в онлайн-середовищі.

Наочний приклад можна розглянути в Таблиці 1.2

Таблиця 1.2 - Переваги та недоліки типів оптимізацій

Тип оптимізації	Характеристики
Оптимізація швидкості завантаження	+ Підвищення SEO-рейтингів + Збільшення конверсій - Складність реалізації - Постійне оновлення
Оптимізація для пошукових систем	+ Збільшення органічного трафіку + Підвищення довіри + Довгострокові результати - Тривалий процес - Конкуренція

Продовження тиблиці 1.2

Тип оптимізації	Характеристики
Мобільна оптимізація	+ Розширення аудиторії + SEO переваги - Додаткові витрати - Складність підтримки різних пристроїв
Оптимізація користувацького досвіду (UX)	+ Підвищення залученості + Зменшення показника відмов - Висока вартість - Часові витрати
Оптимізація контенту	+ Підвищення залученості + Підвищення авторитету - Трудомісткість - Постійні оновлення
Безпека сайту	+ Захист даних + Підвищення довіри - Витрати - Складність
Аналітика та моніторинг	+ Покращене розуміння поведінки користувачів + Прийняття обґрунтованих рішень + Оптимізація маркетингових стратегій - Вартість інструментів - Складність аналізу

Мета обирання типу оптимізації полягає в покращенні конкретних аспектів веб-сайту з метою досягнення стратегічних цілей. Серед даних типів оптимізацій web-застосунків варто обрати оптимізацію щвидкості завантаження. Вона покращить швидкість виконання функцій та прогрузки результатів.

1.3 Постановка задачі

Нехай задано вхідний вектор X(x1, x2, x3, x4), де

x1 – перевірка на наявність 10 гравців.

x2 – надавання імен гравцям.

x3 – надавання ролей гравцям.

x4 – отримання результатів голосування користувачів ведучим.

Тоді, задачу початку гри можна подати у вигляді:

$$F(X) = Y,$$

де Y – вихідний вектор, відображаючий роль, яка перемогла.

1.4 Висновки до розділу 1

Розглянуто існуючі засоби web-застосуноку гри "Мафія", їх можливості та особливості. Методи оптимізації та шляхи покращення web-застосунків. Поставлена задача до бакалаврської дипломної роботи.

2 РОЗРОБКА УДОСКОНАЛЕНОГО АЛГОРИТМУ ДЛЯ ВЕДУЧОГО ГРИ “МАФІЯ”

У більшості існуючих сайтів для гри "Мафія" користувачі стикаються з низкою обмежень, які знижують гнучкість і зручність ігрового процесу. По-перше, ці платформи часто не дозволяють додавати додаткові ролі, що обмежує варіативність гри та можливість експериментувати з новими сценаріями та стратегіями. По-друге, відсутність можливості налаштувати таймер на голосування ускладнює планування і контроль за часом, що може негативно впливати на динаміку гри. Нарешті, більшість платформ не підтримують функцію передчасного завершення голосування, що не дозволяє гнучко реагувати на зміни у грі та приймати оперативні рішення.

Тому, було удосконалено алгоритм для ведучого гри “Мафія” задля покращення функціональності, було створено кнопки таймеру, передчасної зупинки голосування та додавання унікальних ролей.

При цьому було створено функцію таймеру, яка включатиме в себе відведену хвилину часу, щоб гравці могли змогу обговорити та проголосувати в підозрюваного. Функцію зупинення таймеру, яка включатиме в себе можливість достроково завершити голосування та видати результат тих гравців, які брали участь та функцію утворення унікальних ролей, яка включатиме в себе можливість створення спеціальних ролей для цивільних для збільшення можливостей та цікавості гри.

2.1 Розробка удосконаленого узагальненого алгоритмудля ведучого гри “Мафія”.

Тоді удосконалений алгоритм для ведучого гри “Мафія” має такі кроки:

Крок 1: Розпочати гру

Крок 2: Додати гравців

Крок 3: Перевірити чи є 10 гравців

Крок 4: За позитивної відповіді роздати ролі, за негативної додати учасника

Крок 5: За допомогою рандому роздати ролі гравцям

Крок 6: Якщо натиснуто кнопку унікальних ролей роздати нові ролі.

Крок 7: Якщо натиснуто кнопку таймер – розпочати голосування. Голосування завершується у випадку закінчення часу автоматично або самостійно при натисканні кнопки голосувати.

Крок 8: Перевірити чи є ще гравці з роллю мафія. В позитивноній відповіді – продовжити гру, в негативній – вивести результат перемогу цивільних.

Крок 9: Перевірити чи гравців більше ніж 2. У позитивномій відповіді – продовжити грати та розпочати нове голосування, в негативному – завершити гру та вивести результат перемога мафії.

UML-діаграма послідовності удосоконаленого алгоритму роботи гри “Мафія” зображено на рисунку 3.1.

Рисунок 2.1 – UML-діаграма послідовності удосоконаленого алгоритму роботи гри “Мафія”

2.2 Розробка алгоритму вбивства та лікування

Алгоритм вбивства та лікування гравців:

Крок 1: Отримати результат голосування мафії

Крок 2: Отримати результат голосування лікаря

Крок 3: Співставити чи вони проголосували в одного гравця. В позитивному результаті - залишити гравця живим , в негативному видалити гравця проти, якого проголосувала мафія.

UML-діаграми взаємодії алгоритму вбивства та лікування зображено на рисунку 3.2

Рисунок 2.2 – UML-діаграма послідовності алгоритму вбивства та лікування

Додано можливість отримання унікальних ролей, як маньяк та дон. Їх обирають поміж цивільного населення. Дон хоч і являється цивільною роллю, але має можливість вистрілити вночі, перемагає разом з цивільними. Маньяк має особливість вбивати вночі, перемагає у випадку, якщо залишився тільки він.

2.3 Розробка алгоритму додавання унікальних ролей.

Алгоритм отримання унікальних ролей:

Крок 1: Перевірити чи є гравець цивільним, при чому не має ролі лікар або детектив. В позитивній відповіді – за допомогою рандому видати роль маньяк або дон. За негативною відповіддю – вилучити його з варіантів для отримання ролі.

Крок 2: Повідомити про отримання ролі гравця.

UML-діаграма послідовності алгоритму отримання унікальних ролей зображено на рисунку 3.3.

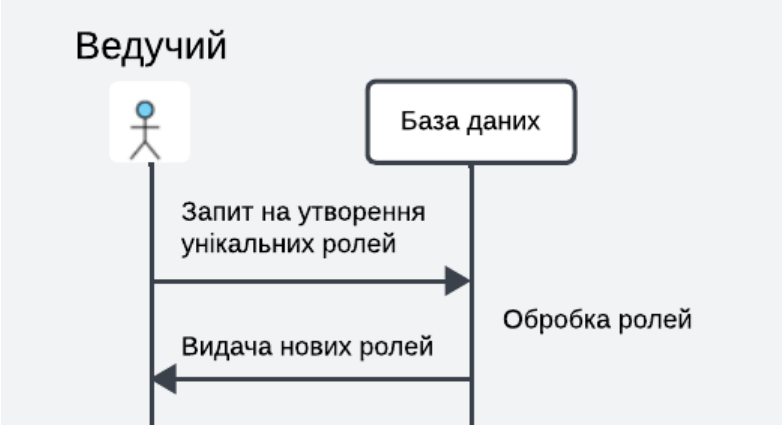


Рисунок 2.3 – UML-діаграма послідовності алгоритму отримання унікальних ролей

2.4 Висновки до розділу 2

Було розроблено алгоритми функцій таймеру, завчасного зупинення голосування, створення унікальних ролей та стурктуру функціонально підвищених завдань для ведучого гри “Мафія”. Побудовано до них UML-діаграми послідовності.

3 РОЗРОБКА СТРУКТУРИ WEB-ЗАСТОСУНКУ ДЛЯ ГРИ “МАФІЯ”

3.1 Розробка блоків web-застосунку

Для реалізації удосконаленого алгоритму для ведучого гри “Мафія” , відповідний WEB-застосунок повинен містити такі блоки:

1. Блок перевірки кількості гравців, який перевірятиме на наявність мінімальної кількості гравців.
2. Блок управління ролями, задачою якого є роздача ролей.
3. Блок оцінки стану гри, задачою якого є обробка інформаціх про колькість гравців та виведення результату при завершені гри.
4. Блок управління функціями, ціллю якого є управління новими функціями для удосконалення функціональних можливостей.

Блок перевірки кількості гравців взаємодіє з блоком управління ролями розпочинаючи гру за достатньою кількості гравців та дає можливість блоку управління ролями роздати ролі гравцям. Блок управління ролями взаємодіє з блоком оцінки стану гри, надаючи інформацію про кількість гравців з роллю мафія, за відсутністю якої гра завершиться. Блок управління функціями взаємодіє з іншими блоками, адже підтримуватиме функціональні можливості за негативної відповіді блоку оцінки стану гри та працюватиме за позитивної відповіді блоку перевірки кількості гравців.

Тоді UML-діаграма взаємодії блоків структури WEB-застосунку для ведучого гри “Мафія” має вигляд представлений на рисунку 3.1.

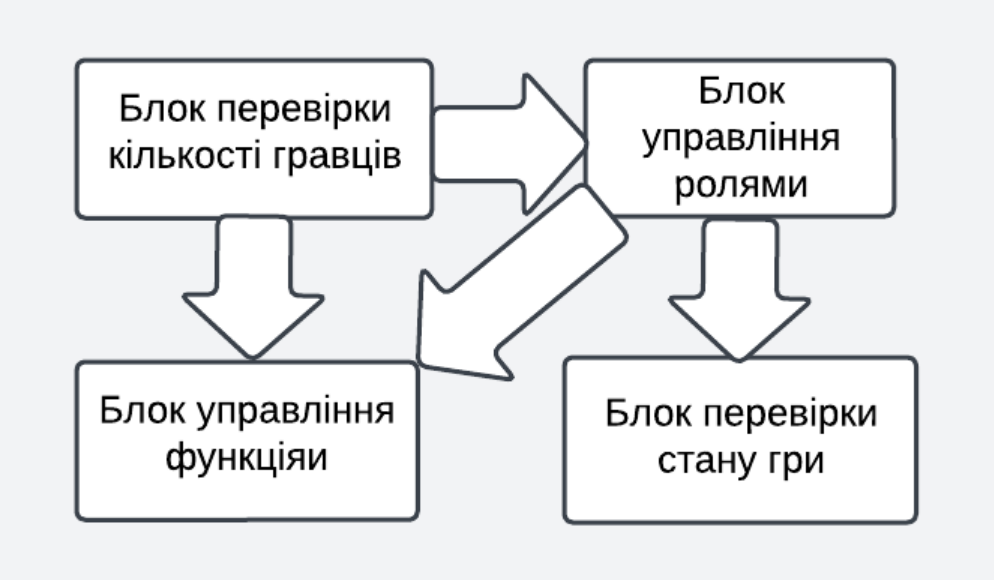


Рисунок 3.1 - UML-діаграма взаємодії блоків структури WEB-застосунку для ведучого гри “Мафія”

3.2 Висновки

Отож, запропоновано блоки структури web-застосунку, які забезпечать розширення функціональних можливостей для ведучого гри “Мафія”, які опрацьовують створені функції та керують їх взаємодією.

4 РОЗРОБКА СКЛАДОВИХ WEB-ЗАСТОСУНКУ ДЛЯ ВЕДУЧОГО ГРИ “МАФІЯ”

4.1 Вибір мови програмування

Успішна розробка веб-застосунку значною мірою залежить від правильного вибору мови програмування. Це рішення впливає на продуктивність, масштабованість, зручність підтримки та інтеграції з іншими системами. У рамках даного проекту, присвяченого розробці веб-застосунку для ведучого гри “Мафія”, було обрано мову програмування JavaScript. У цьому розділі розглянуто причини вибору саме цієї мови, її переваги та недоліки, а також альтернативні варіанти [14].

Причини вибору JavaScript:

JavaScript є однією з найпопулярніших мов програмування у веб-розробці. Основні причини вибору JavaScript для цього проекту включають:

Широке використання та підтримка: JavaScript використовується практично на всіх сучасних веб-сайтах та підтримується всіма популярними браузерами, що робить його незамінним інструментом для фронтенд-розробки.

Висока продуктивність: Завдяки двигунам JavaScript, таким як V8 (який використовується в Google Chrome і Node.js), JavaScript має високу продуктивність, що дозволяє створювати швидкі та ефективні веб-застосунки.

Широкий екосистема: Велика кількість бібліотек і фреймворків (React, Angular, Vue.js) дозволяють швидко розробляти складні веб-застосунки, використовуючи готові рішення та знижуючи час розробки.

Універсальність: JavaScript можна використовувати як для клієнтської, так і для серверної частини завдяки Node.js, що дозволяє створювати повноцінні веб-застосунки на одній мові програмування.

Активна спільнота та підтримка: JavaScript має одну з найбільших та найактивніших спільнот розробників, що забезпечує постійну підтримку, розвиток та наявність великої кількості ресурсів для навчання та вирішення проблем.

Переваги JavaScript:

Кросплатформеність: JavaScript працює у всіх сучасних браузерах і не залежить від операційної системи користувача.

Динамічність: Мова є динамічно типізованою, що дозволяє більш гнучко працювати з даними та швидко реагувати на зміни у вимогах проекту.

Асинхронність: JavaScript підтримує асинхронне програмування, що є важливим для розробки швидких і ефективних веб-застосунків.

Швидке прототипування: Завдяки великій кількості готових бібліотек та фреймворків, JavaScript дозволяє швидко створювати прототипи та експериментальні проекти.

Недоліки JavaScript:

Безпека: JavaScript-код, що виконується на стороні клієнта, є вразливим до атак, таких як XSS (міжсайтовий скриптинг). Вимагає додаткових заходів безпеки для захисту даних.

Відсутність типізації: Відсутність статичної типізації може призвести до помилок, які важко відстежувати та виправляти.

Проблеми сумісності: Хоча більшість сучасних браузерів підтримують JavaScript, існують деякі відмінності у реалізації, що може викликати проблеми сумісності.

Альтернативні варіанти

При виборі мови програмування для веб-застосунку можна розглянути також інші мови, такі як Python (з використанням фреймворків Django або Flask), Ruby (з фреймворком Ruby on Rails) або PHP. Кожна з цих мов має свої переваги та недоліки:

Python: Легко читається і пишеться, має велику кількість бібліотек, але може бути повільнішою у виконанні порівняно з JavaScript.

Ruby: Зручний для швидкої розробки завдяки фреймворку Ruby on Rails, але має меншу спільноту та екосистему порівняно з JavaScript.

PHP: Популярний для серверної частини, легко інтегрується з базами даних, але менш гнучкий для фронтенд-розробки.

Порвінянн мов програмування зображено в таблиці 4.1.

Таблиця 4.1 – Порівняння мов програмування

Мова програмування	Безпека	Легкість розуміння	Розноманіття бібліотек	Швидкість роботи
PHP	-	-	-	-
Ruby	-	+	-	-
Python	+	+	+	-
JavaScript	+	+	+	+

Вибір мови програмування є критичним етапом у розробці веб-застосунку. JavaScript був обраний для цього проекту через його універсальність, високу продуктивність, велику екосистему та активну спільноту. Використання JavaScript дозволяє ефективно реалізувати всі необхідні функції веб-застосунку для ведучого гри “Мафія”, забезпечуючи високий рівень продуктивності, зручності та безпеки для користувачів.

4.2 Вибір середовища програмування та проектування класів

При розробці веб-застосунку важливим етапом є вибір середовища програмування (IDE), яке забезпечує зручність і ефективність процесу розробки. Для даного проекту, спрямованого на створення веб-застосунку для ведучого гри “Мафія”, було обрано Visual Studio Code (VS Code). У цьому розділі буде розглянуто причини вибору саме цього середовища, його переваги та недоліки, а також альтернативні варіанти[15].

Причини вибору Visual Studio Code[16]

Visual Studio Code (VS Code) є одним з найпопулярніших і найпотужніших середовищ програмування для веб-розробки. Основні причини вибору VS Code для цього проекту включають:

Легка вага та швидкість: VS Code є легким та швидким середовищем, що забезпечує оперативний запуск та роботу навіть на менш потужних комп'ютерах.

Багатофункціональність: VS Code підтримує широкий спектр мов програмування та технологій завдяки великій кількості розширень, що дозволяє ефективно працювати з JavaScript, HTML, CSS та іншими технологіями, необхідними для розробки веб-застосунків.

Інтеграція з Git: Вбудована підтримка Git полегшує керування версіями коду та забезпечує зручну інтеграцію з репозиторіями, що є важливим для командної розробки та контролю змін.

Розширюваність: Велика кількість розширень та плагінів дозволяють налаштовувати середовище під специфічні потреби розробника, додаючи необхідні функції та інструменти.

Зручний інтерфейс: Інтерфейс VS Code є інтуїтивно зрозумілим та зручним, що сприяє швидкому освоєнню та підвищенню продуктивності роботи.

Безкоштовність та відкритий код: VS Code є безкоштовним інструментом з відкритим кодом, що робить його доступним для широкого кола розробників.

Переваги Visual Studio Code

Підтримка різних мов програмування: Окрім JavaScript, VS Code підтримує багато інших мов, що робить його універсальним інструментом для розробки.

Велика кількість розширень: Маркетплейс розширень надає доступ до багатьох корисних інструментів, таких як eslint, prettier, live server та інших.

Інтеграція з терміналом: Вбудований термінал дозволяє виконувати командний рядок безпосередньо в середовищі розробки, що спрощує роботу з різними інструментами та командами.

Постійні оновлення та підтримка: VS Code активно розвивається та підтримується спільнотою та компанією Microsoft, що забезпечує регулярні оновлення та поліпшення функціоналу.

Недоліки Visual Studio Code

Споживання ресурсів: Хоча VS Code є легким у порівнянні з іншими IDE, він все ж може споживати значну кількість ресурсів при використанні великої кількості розширень.

Налаштування: Велика кількість налаштувань та опцій може заплутати новачків, що потребує додаткового часу для освоєння.

Обмеженість базових можливостей: Без встановлення додаткових розширень базовий функціонал може бути обмеженим для деяких специфічних задач.

Альтернативні варіанти

Серед альтернативних середовищ програмування, які також можуть бути використані для розробки веб-застосунків, можна виділити:

Atom[17]: Відкритий текстовий редактор від GitHub, який має схожі можливості з VS Code, але може бути менш швидким.

Sublime Text[18]: Легкий та швидкий текстовий редактор з підтримкою плагінів, але не має стільки інтегрованих можливостей, як VS Code.

WebStorm[19]: Платне середовище від JetBrains, спеціалізоване на JavaScript та інших веб-технологіях, пропонує багатий функціонал та інтеграцію, але вимагає фінансових витрат.

Brackets[20]: Легкий редактор з відкритим кодом, орієнтований на веб-розробку, але має менш активну спільноту порівняно з VS Code.

Вибір середовища програмування є важливим етапом у розробці веб-застосунку. Visual Studio Code був обраний для цього проекту через його легкість, багатофункціональність, розширюваність та зручність використання. Використання VS Code дозволяє ефективно реалізувати всі необхідні функції веб-застосунку для ведучого гри “Мафія”, забезпечуючи високий рівень продуктивності, зручності та безпеки для розробників.

Порівняння середовищ розробки зображено в таблиці 4.2

Таблиця 4.2 – порівняння середовищ розробки

Характеристика	Visual Studio Code	Atom	Sublime Text	WebStorm	Brackets
Платформи	Windows, Mac, Linux	Windows, Mac, Linux	Windows, Mac, Linux	Windows, Mac, Linux	Windows, Mac, Linux
Мова розширень	JavaScript, TypeScript	JavaScript, TypeScript	Python	JavaScript	JavaScript
Підтримка плагінів	Так	Так	Так	Так	Так
Швидкість роботи	Висока	Середня	Висока	Висока	Середня
Налаштовуваність	Висока	Висока	Висока	Висока	Висока

Отже, було обрано середовище розробки Visual Studio Code, через пыдтримку рызноманытних плагінів,, бібліотек та підтримку взаємодії між мовами програмування.

4.3 Розробка інтерфейсу web-застосуноку

Структура web-застосунку гри “Мафія” для ведучого матиме такий вигляд:

Поділений на 4 частини екран для відображення різних частин функціоналу сайту:

1. Користувацька частина. Будуть створенні кнопки додавання гравців, початку гри та утворення унікальних ролей.
2. Частина відображення гравців. Будуть відображені всі гравці, їх стан чи живий, чи мертвий та їх ролі.
3. Частина голосування. Будуть відображені поля, в яких будуть обирати за кого голосувати.

4. Частина таймеру. Будуть додані кнопки для початку таймера та завчасного призупинення голосування.

Структуру інтерфейсу зображено на рисунку 3.1.

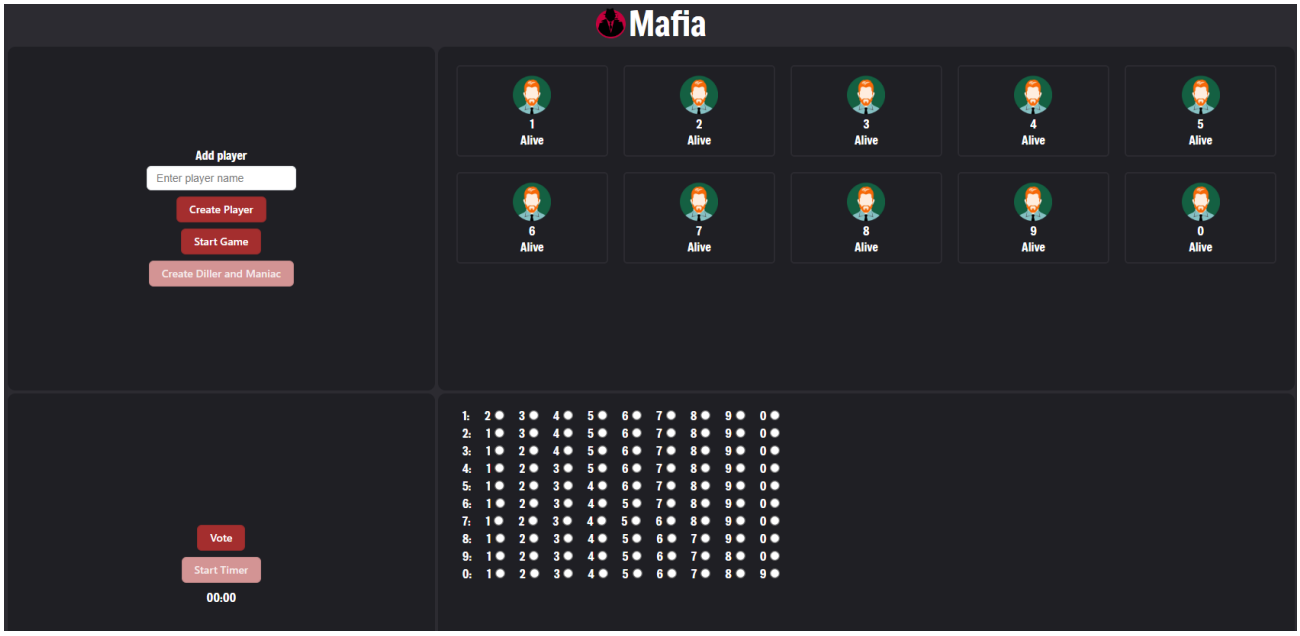


Рисунок 4.1 – корисутувацький інтерфейс

Для розробки інтерфейсу сайту було використано CSS.

Створено оформлення для фону сайту, заставок гравців, кнопок.

Налаштування фону, положення заставок гравців, кнопок голосування та поля голосування(рис.4.2):

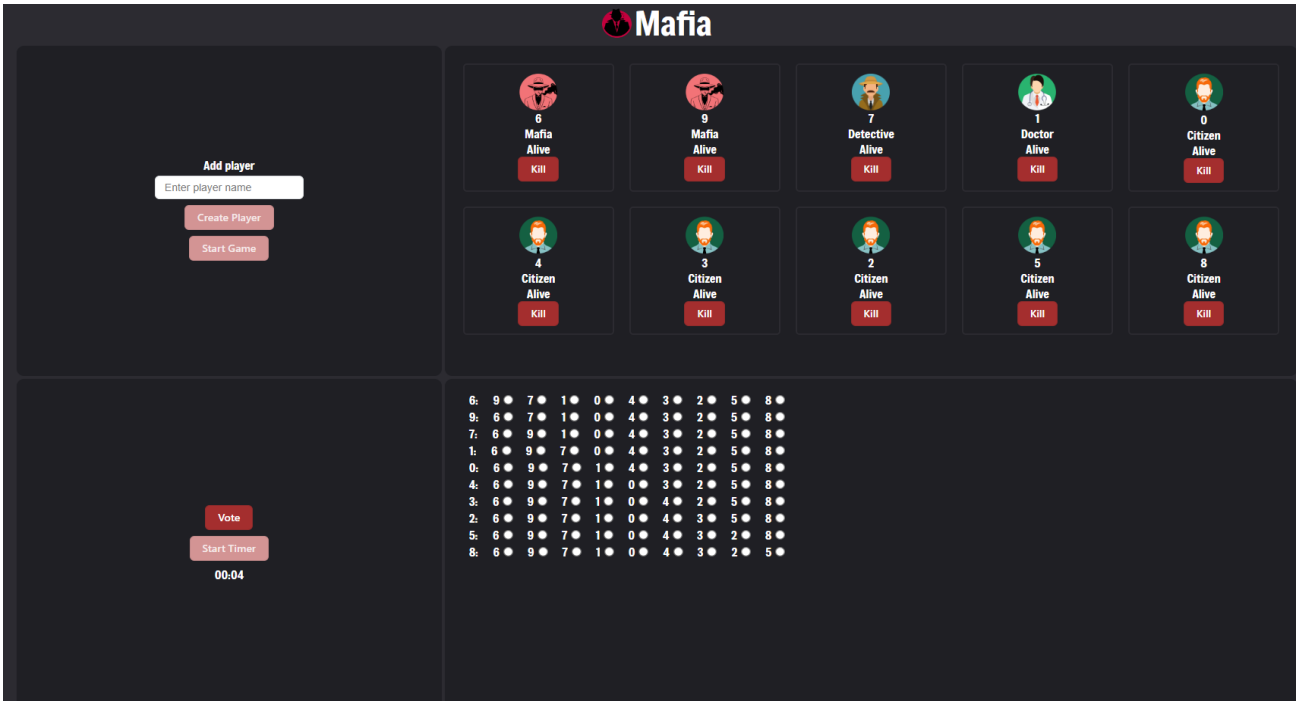


Рисунок 4.2 – Оформлення фону та положення атрибутів сайту

Налаштування іконок ролей та гравців(рис.4.3):

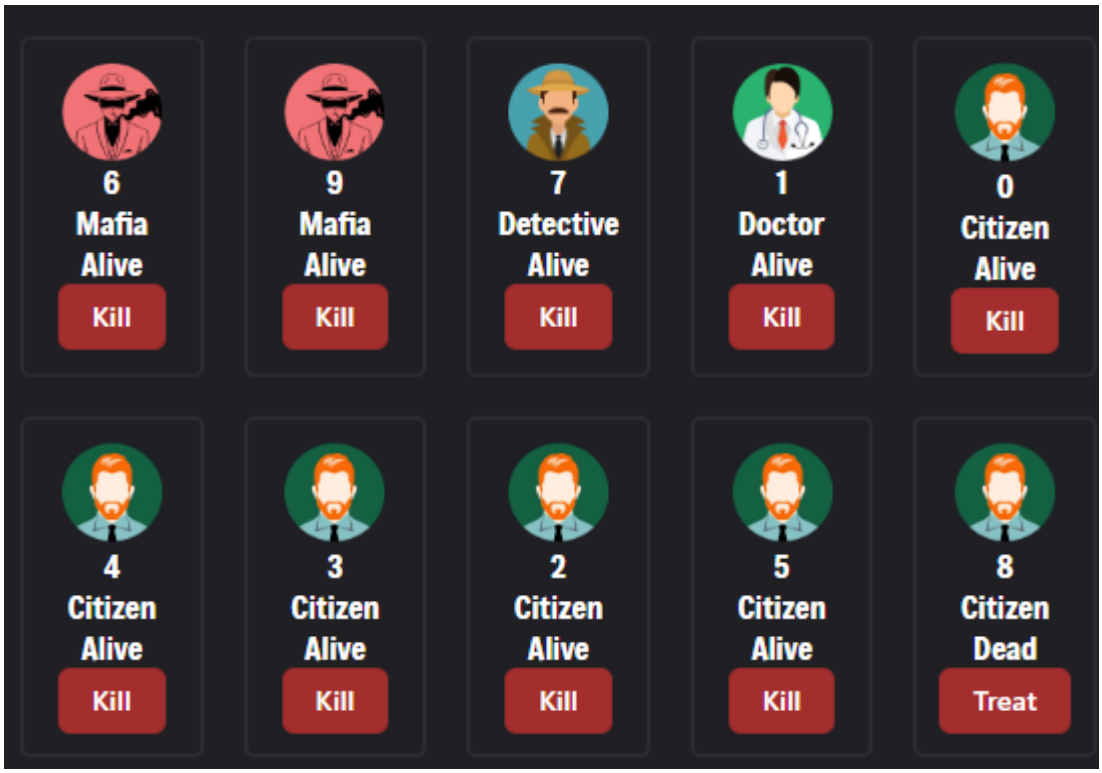


Рисунок 4.3 – оформлення іконок ролей та гравців

#create-player-input {

```
    padding: 5px 12px;
    font-size: 14px;
    line-height: 20px;
    color: #24292e;
    vertical-align: middle;
    background-color: #ffffff;
    background-repeat: no-repeat;
    background-position: right 8px center;
    border: 1px solid #e1e4e8;
    border-radius: 6px;
    outline: none;
    box-shadow: rgba(225, 228, 232, 0.2) 0px 1px 0px 0px inset;
    :focus{
        border-color: #d60303;
        outline: none;
        box-shadow: rgba(214, 3, 3, 0.3) 0px 0px 0px 3px;
    }
}
```

Налаштування кнопок(рис4.4-4.7):

Створені кнопки створення гравця, початку гри та надання унікальних ролей(рис.4.3) та текстове поле для введення ім’я гравця.

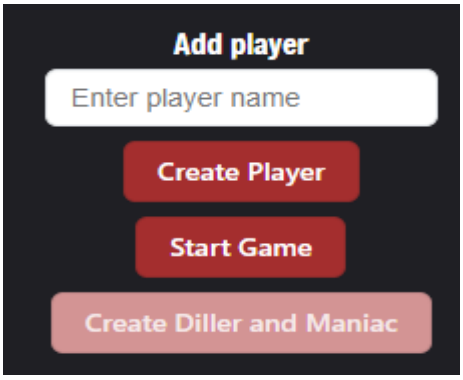


Рисунок 4.4 – кнопки додавання гравців та початку гри

```
function createPlayer (name, role) {
    playerList.push(new Player(name, role))
    console.log(`Player ${name} created.`)
    console.log(playerList)
}
const startGameButton = document.createElement('button')
startGameButton.textContent = 'Start Game'
startGameButton.classList.add('button-3')
startGameButton.role = 'button'
```

Створено кнопки початку таймера та завчасного зупинення голосування(рис 4.4). Кнопка таймеру розпочинає відлік 1 хвилини та по завершенню видає результат голосування. Кнопка “Vote” завчасно зупиняэ таймер та видаэ результат тих гравцыв, якы вспыли проголосувати.

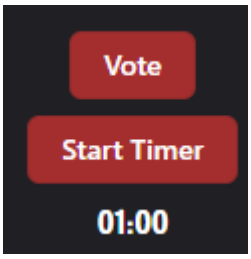


Рисунок 4.5 – кнопки початку таймера та призупинення голосування

```
const voteButton = document.createElement('button')
voteButton.textContent = 'Vote'
voteButton.classList.add('button-3')
voteButton.role = 'button'


playerWithMostVotes.die()
  clearInterval(intervalId); // Stop the timer
  intervalId = null;
  startTime = null;
  timerButton.disabled = false; // Enable the button again
```

```
  timerDisplay.textContent = '00:00';
  voting = false
  updatePlayerList()
})
```

Створено кнопку вбивства гравця. Яка виганяє гравця та забирає можливість голосувати(рис. 4.6).

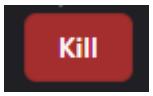


Рисунок 4.6 – кнопка “вбити гравця”

```
if (player.alive && gameStarted) {
    // If player is alive create kill button
    const killButton = document.createElement("button")
    killButton.textContent = "Kill"
    killButton.style.display = "inline-block"
    killButton.classList.add("button-3")
    killButton.addEventListener("click", () => {
      player.die()
      updatePlayerList()
    })
```

Створена кнопка функції лікування гравця, яка повертає гравцю можливість голосуватию(рис.4.7)

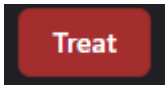


Рисунок 4.7 – кнопка “вилікувати гравця”

```
    treat() {
        this.alive = true
        console.log(`${this.name} is treated.`)
        updatePlayerList()
```

```
  }
}
```

Створення ролей діллера та маньяка створенна окрема кнопка(рис.4.8). Яка обирає з поміж цивільних два гравця і випадковим чином дає їх 2 унікальні ролі.

Рисунок 4.8 – кнопка створення діллера та маньяка

```
dillerAndManiacButton.addEventListener('click', () => {
  playerList[4].role = 'Diller'
  playerList[5].role = 'Maniac'
  updatePlayerList()
  dillerAndManiacButton.disabled = true
})
```

4.4 Тестування функціонування web-застосунку для ведучого гри “Мафія”

Для порівняння швидкодії реагування сайтів на зміни було взята середні значення 100 перевірок.(таблиця 4.1)

Таблиця 4.1 – середній результат тестування відгуку на зміни

Назва сайту	Середнє значення часу відгуку (с)
Розроблений сайт на БКР	0.021
Mafia GG	0.034
Mafia: The Game	0.027

Тестування кнопок запуску таймерата завчасного припинення голосування. При натисканні на кнопку початку таймера(рис. 4.9)

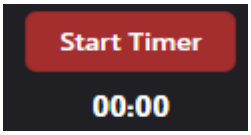


Рисунок 4.9 – кнопка початку таймера

З'являється поле для голосування та починається відлік 60 секунд, при сплину часу результати голосування автоматично виводяться та виганяється гравець з найбільшою кількістю голосів.(рис.4.10, рис.4.11)

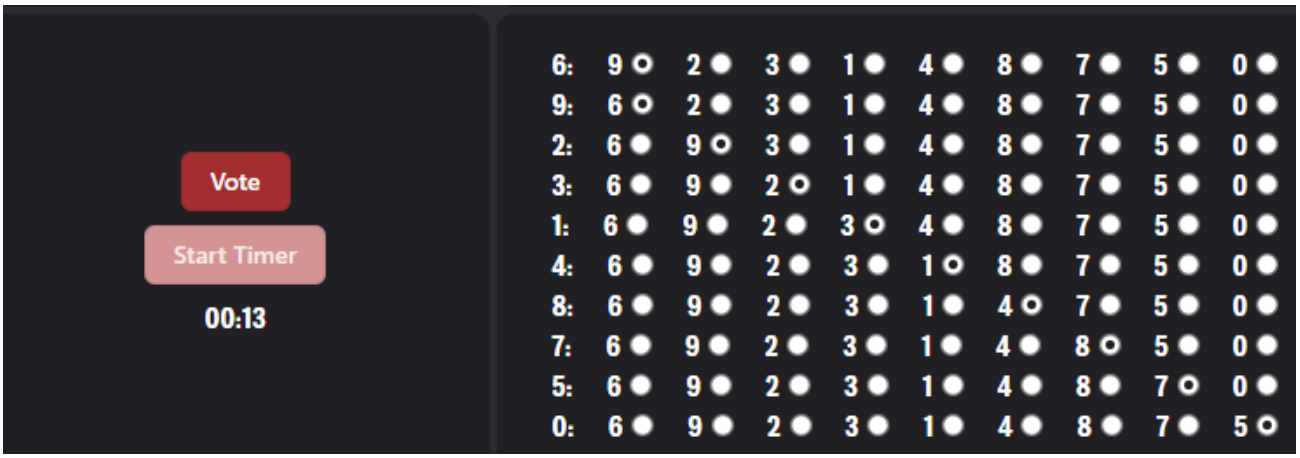


Рисунок 4.10 – поле для голосування та таймер

Рисунок 4.11 – вигнаний гравець

При натисканні на кнопку голосувати, таймер завчасно зупиняється та виганяється гравець з найбільшою кількістю голосів.(рис.4.12, рис.4.10)

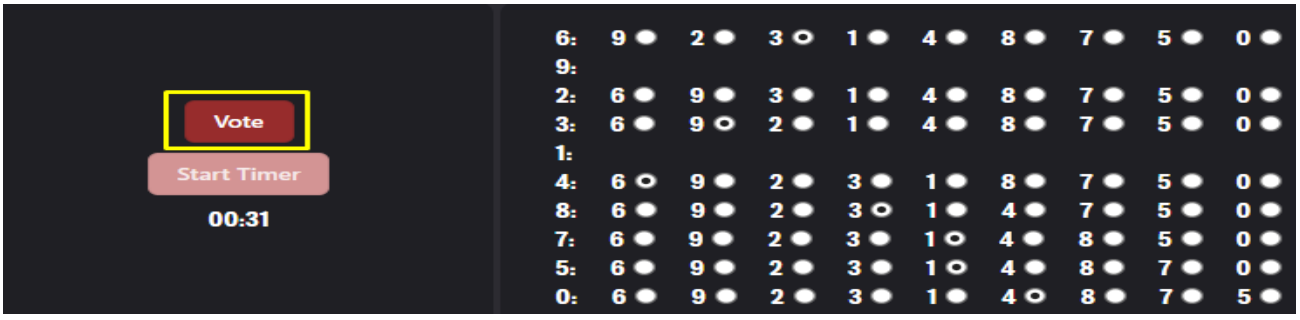


Рисунок 4.12 – кнопка голосувати та її дія

Вигнаний або вбитий мафією гравець втрачає можливість до голосування.

Тестування кнопки створення унікальних ролей: діллер та маньяк.

На рисунку 4.13 зображено ролі гравців до натискаання на кнопку утворення унікальних ролей. Є 2 мафії, 1 лікар, 1 детектив та 6 цивільних. На рисунку 4.14 зображено ролі гравців та отримані нові унікальні ролі. Є 2 мафії, 1 лікар, 1 шериф, 1 дон, 1 маньяк та 4 цивільних.

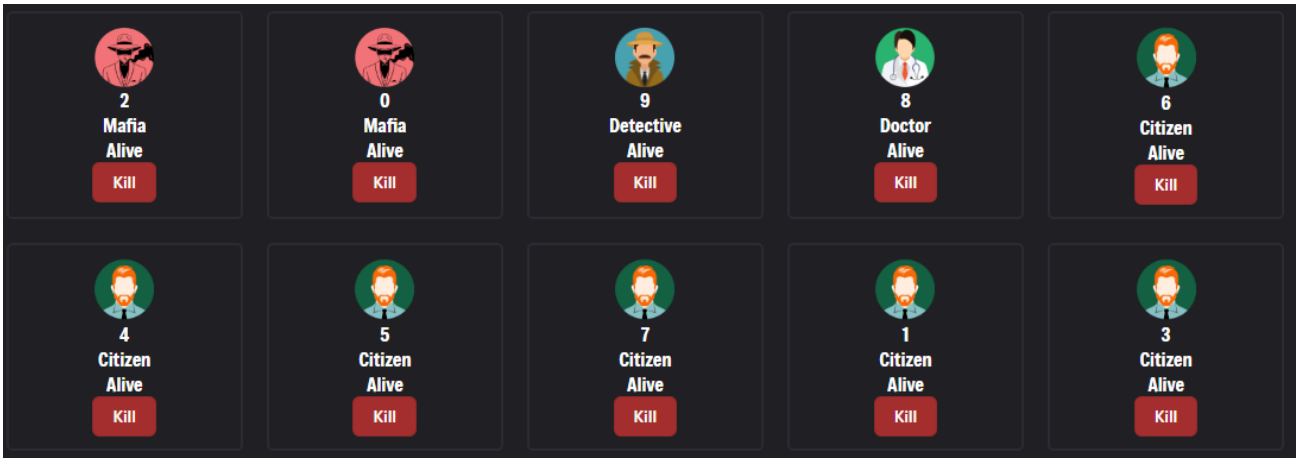


Рисунок 4.13 – ролі гравців до натискаання на кнопку утворення унікальних ролей.

Рисунок 4.14 – ролі після натискання на отримання унікалних ролей

4.5 Висновки до розділу 4

Було розглянуто реалізацію сайту для ведучого гри “Мафія”. Описано, як працюють створені кнопки, як описаний інтерфейс. Обгрунтовано вибір мови програмування та середовища реалізації. Протестовано правильність виконання нових функцій.

5 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

5.1 Порівняння з існуючими аналогами

В першому розділі було обрано кращий з запропонованих web-застосунків Mafia: The Game. В цьому розділі буде показано удосконалення функціональних можливостей розробленого web-застосунку.

Додана функція таймеру для голосування.

На web-застосунку Mafia: The Game підчас голосування з’являється вікно зі списком гравців(рис. 5.1). Обравши гравця, змінити голос стає неможливим.

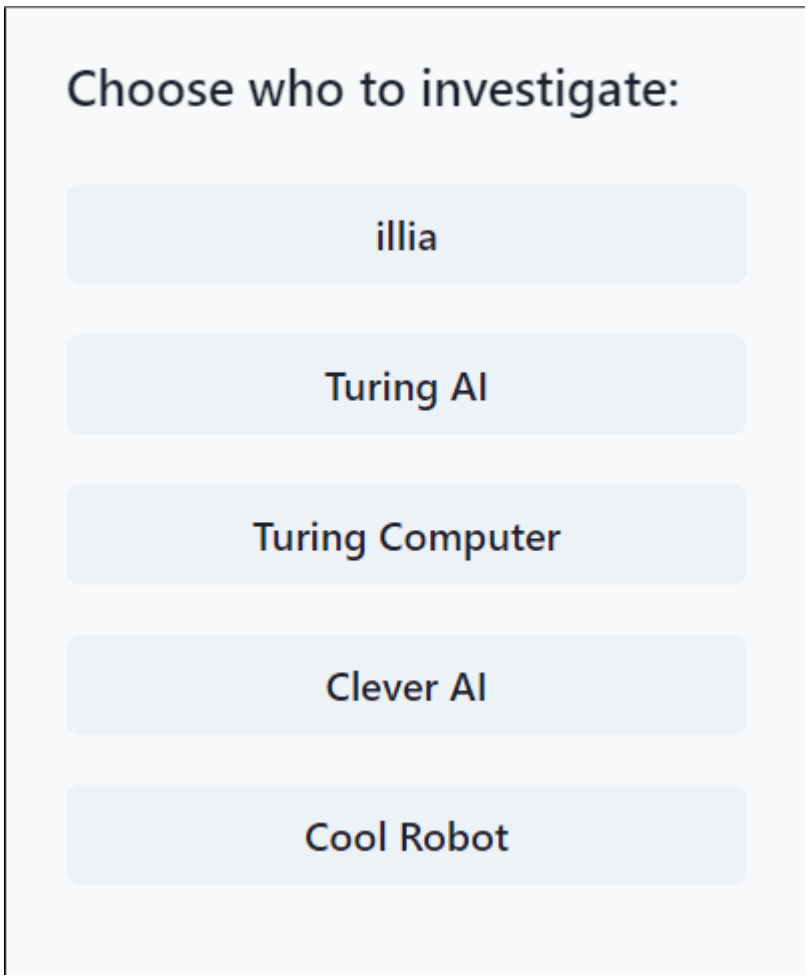


Рисунок 5.1 – список гравців на голосуванні

В розробленому web-застосунку створений таймер на 60 секунд, який дає змогу маніпулювати голосами на протязі даного часу.(рис. 5.2-5.3)

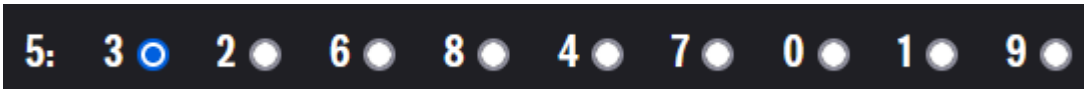


Рисунок 5.2 – початкові голоси

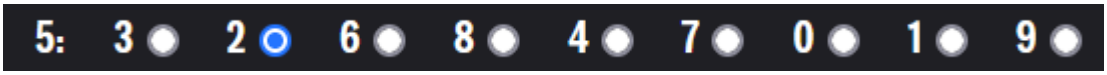


Рисунок 5.3 – змінені голоси

Також додана функція створення унікальних ролей, таких як діллер та маньяк.

У web-застосунку Mafia: The Game є 4 ролі: мафія, шериф, лікар та цивільний. В розробленому web-застосунку є можливість за потреби додати, ще дві ролі: діллер та маньяк.(рис. 5.4-5.5). Отже є 6 ролей: мафія, лікар, шериф, цивільний та унікалі ролі маньяк та діллер.

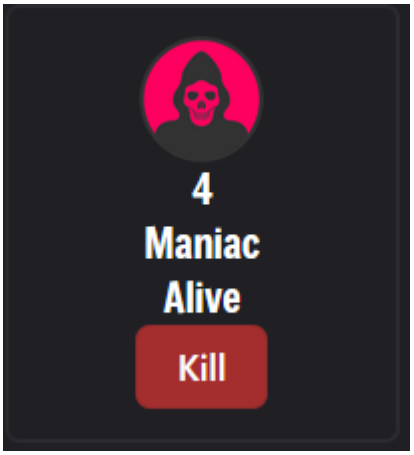


Рисунок 5.4 – роль маньяк

Рисунок 5.5 – роль діллера

При запуску гри з 9 гравцями виведеться таке повідомлення(рис 5.6):

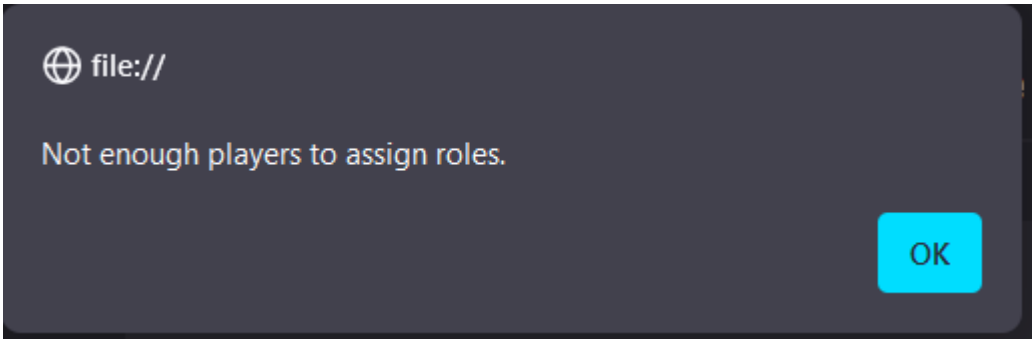
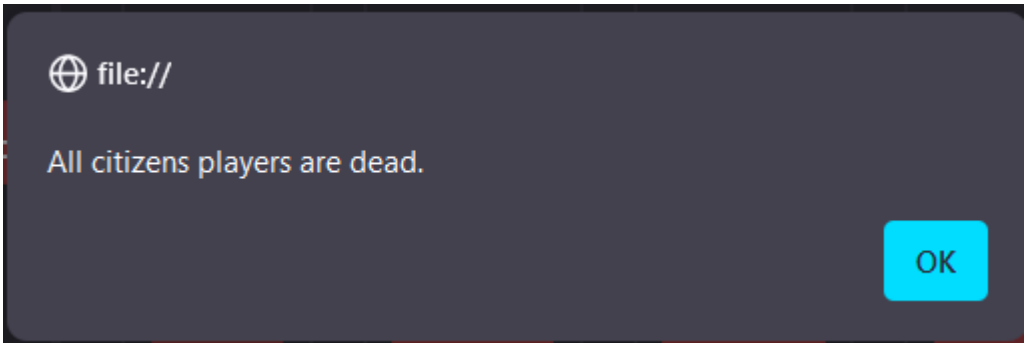


Рисунок 5.6 – повідомленя при запуску з 9 гравцями

Якщо ж додати 10 гравця, то гра розпочнеться(рис.5.7).

Рисунок 5.7 – початок гри

За ситуації коли всі цивільні вбиті, тобто мафія виграла виведеться вікно на рисунку 5.8.



Риснок 5.8 – результат гри

Та у випадку вигнання всіх гравців мафії буде результат, як на рисунку 5.9.

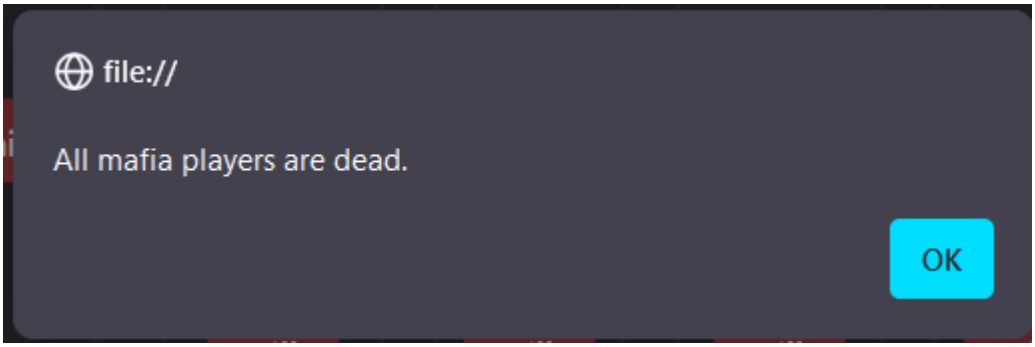


Рисунок 5.9 – результат гри

Отже, за результатами експериментального дослідження впрорівнянні з обраним засобом функціональні можливості розширились за рахунок введення додаткових функцій, які подані в таблиці 5.1

Таблиця 5.1 – Порівняння функціональності

Функціональні можливості	Mafia: The Game	Розроблений web-застосунок
Таймер	-	+
Завчасне зупинення таймеру	-	+
Додавання унікальних ролей	-	+

5.2 Висновки до розділу 5

Було порівняно розроблений web-застосунок з існуючим аналогом, який було обрано в першому розділі, показано переваги розробленого продукту та різницю в функціональних можливостях.

ВИСНОВОКИ

Всі роботи, поставлені перед бакалаврською дипломною роботою, виконані в повному об’ємі, а саме:

- зроблено аналіз існуючих засобів web-сайтів гри “Мафія”

- спроектовано складові частини вебресурсу

- реалізовано нові функціональні можливості сайту

Вхідні данні , а саме: 10 гравців, 2 мафії, як найменьше 1 голос в голосуванні, виконують свою роль для виведення результату.

В ході бакалаврської дипломної роботи були проаналізовані найпопулярніші сайти для гри “Мафія”, було вказано їх переваги та недоліки. Експериментально порівняно web-застосунок для виконання БДР з найкращим з переглянутих сайтів.

Було проаналізовано різні типи структур вебресурсу, різні види оптимізацій сайтів та розроблено покращений алгоритми функціонування частин вебресурсу.

Відповідно з завданням до бакалаврської дипломної роботи було розроблені нові функціональні можливості до сайту, а саме: тамер, завчасне припинення голосування та створення унікальних ролей. Також за допомогою автоматизованих тестів було проведено тестування швидкодії.

Мета роботи – розширення функціональних можливостей web-застосунку гри “Мафія” для ведучого досягнута за рахунок того, що в порівнянні з аналогами введені додаткові функції, а саме було додано тамер, завчасне припинення голосування та створення унікальних ролей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Розробка удосконаленого алгоритму для ведучого гри “Мафія”. URL:
2. Розробка структури web-застосунку для ведучого гри “Мафія”. URL:
3. Mafia: The Game [Електронний ресурс] – Режим доступу: https://mafiathegame.benrobinson.dev
4. Polemica [Електронний ресурс] – Режим доступу: https://polemicagame.com
5. MafiaGG [Електронний ресурс] – Режим доступу: https://mafia.gg
6. Оптимізація сайту [Електронний ресурс] – Режим доступу: https://ideadigital.agency/blog/seo-optimizacziya-sajtu-samostiino/
7. Оптимізація швидкості завантаження [Електронний ресурс] – Режим доступу: https://project-seo.net/blog-uk/optimizaciya-shvidkosti-zavantazhennya-korporativnih-sajtiv/
8. Оптимізація пошукових систем [Електронний ресурс] – Режим доступу: https://bevisible.com.ua/blog/scho-take-seo/
9. Мобільна оптимізація [Електронний ресурс] – Режим доступу: https://info.nic.ua/uk/blog-uk/optymizacziya-mobilnyh-prystroyiv/
10. Оптимізація безпеки сайту [Електронний ресурс] – Режим доступу: https://www.linkedin.com/pulse/9-дієвих-тактик-з-безпеки-сайту-без-навичок-роман-манзенко-syhre?trk=article-ssr-frontend-pulse_more-articles_related-content-card
11. Оптимізація користувацького досвіду [Електронний ресурс] – Режим доступу: https://www.ranktracker.com/uk/blog/user-experience-and-conversion-rate-optimization-key-principles-for-success/
12. Оптимізація контенту [Електронний ресурс] – Режим доступу: https://interfax.com.ua/news/press-release/925120.html
13. Аналітика і моніторинг [Електронний ресурс] – Режим доступу: https://hostiq.ua/blog/ukr/35-tools-for-web-analytics/

14. Вибір мови програмування [Електронний ресурс] – Режим доступу: https://foxminded.ua/movy-prohramuvannia-dlia-stvorennia-saitiv/

15. Середовища програмування [Електронний ресурс] – Режим доступу: https://ua5.org/osnprog/1600-populyarni-seredovyshha-rozrobky-mov-programuvannya.html

16. Visual Studio Code [Електронний ресурс] – Режим доступу: https://visualstudio.microsoft.com/ua/

17. Atom [Електронний ресурс] – Режим доступу: https://atom-editor.cc

18. SublimeText [Електронний ресурс] – Режим доступу: https://www.sublimetext.com

19. WebStorm [Електронний ресурс] – Режим доступу: https://www.jetbrains.com/ua-ua/webstorm/

20. Brackets [Електронний ресурс] – Режим доступу: https://brackets.io

ДОДАТОК А
ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка WEB-застосунку для ведучого гри «Мафія»

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 95,8% Схожість 4,2%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку Озеранський В.С.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи Доценко І.К.

Керівник роботи Савчук Т.О.

Додаток Б

Лістинг програми

```
// classes and functions


// List of available roles
const availableRoles = ['Citizen', 'Doctor', 'Detective', 'Mafia', 'Maniac', 'Diller']
const rolesList = ["Mafia", "Mafia", "Detective", "Doctor", "Citizen", "Citizen", "Citizen", "Citizen", "Citizen", "Citizen",]


// Check if all mafia players are dead
function areAllMafiaDead() {
  const mafias = playerList.filter(player => {
    return player.role === 'Mafia' && player.alive
  })
  if (mafias.length < 1) {
    return true
  } else {
    return false
  }
}


// Check if all except mafia are dead
function areAllCitizensDead() {
  const exceptMafias = playerList.filter(player => {
    return player.role !== 'Mafia' && player.alive
  })
```

```
  if (exceptMafias.length < 1) {
    return true
  } else {
    return false
  }
}

// Player class
class Player {
  constructor(name, role) {
    this.name = name
    this.role = role
    this.alive = true
    this.votes = 0
  }

  // Method to select player`s role
  selectRole(role) {
    this.role = role
    console.log(`${this.name} selected role: ${this.role}`)
  }

  // Method to making this player dead
  die() {
    this.alive = false
    console.log(`${this.name} died.`)
```

```
    updatePlayerList()
  }

  // Method to making this player alive again
  treat() {
    this.alive = true
    console.log(`${this.name} is treated.`)
    updatePlayerList()
  }
}

// Function to create new player
function createPlayer (name, role) {
    playerList.push(new Player(name, role))
    console.log(`Player ${name} created.`)
    console.log(playerList)
}

// Function to update the HTML player list
function updatePlayerList() {
  // Check if all mafia is dead
  if (areAllMafiaDead() && gameStarted) {
      console.log('All mafia players are dead.')
      alert('All mafia players are dead.')
      location.reload()
      return
```

```
}

// Check if all except mafia are dead
if (areAllCitizensDead() && gameStarted) {
    console.log('All except mafia players are dead.')
    alert('All citizens players are dead.')
    location.reload()
    return
}

playerListElement.innerHTML = '' // Clear existing list items
votesHtmlList.innerHTML = '' // Clear existing list items
playerList.forEach(player => {
  // Create list item for each player
  const listItem = document.createElement('li')

  // Create player icon element
  const playerIcon = document.createElement('img')
  switch (player.role) {
    case "Mafia":
      playerIcon.src = "icons/mafia.png"
      break;
    case "Detective":
      playerIcon.src = "icons/detective.png"
      break;
    case "Doctor":
```

```
    playerIcon.src = "icons/doctor.png"
    break;
  case "Citizen":
    playerIcon.src = "icons/citizen.png"
    break;
  case "Maniac":
    playerIcon.src = "icons/maniac.png"
    break;
  case "Diller":
    playerIcon.src = "icons/dealer.png"
    break
  default:
    playerIcon.src = "icons/citizen.png"
    break;
}
playerIcon.classList.add('player-icon')

// Create player name span element
const playerName = document.createElement('span')
playerName.classList.add('player-name')
playerName.textContent = player.name

// Create player role span element
const playerRole = document.createElement('span')
playerRole.classList.add('player-role')
playerRole.textContent = player.role
```

```
    // Create player status span element
    const playerStatus = document.createElement('span')
    playerStatus.classList.add('player-status')
    playerStatus.textContent = player.alive ? 'Alive' : 'Dead'

    // listItem.innerHTML = `${player.name}: ${player.role}; ${player.alive ? 'Alive' :
'Dead'}`

    // Append all elements into listItem
    listItem.appendChild(playerIcon)
    listItem.appendChild(playerName)
    listItem.appendChild(playerRole)
    listItem.appendChild(playerStatus)

    // Append listItem to playerListElement
    playerListElement.appendChild(listItem)

    if (player.alive && gameStarted) {
      // If player is alive create kill button
      const killButton = document.createElement("button")
      killButton.textContent = "Kill"
      killButton.style.display = "inline-block"
      killButton.classList.add("button-3")
      killButton.addEventListener("click", () => {
        player.die()
        updatePlayerList()
```

```
  })
  listItem.appendChild(killButton)
} else if (!player.alive && gameStarted) {
  // If player is`t alive create treat button
  const treatButton = document.createElement("button")
  treatButton.textContent = "Treat"
  treatButton.style.display = "inline-block"
  treatButton.classList.add("button-3")
  treatButton.addEventListener("click", () => {
    player.alive = true
    updatePlayerList()
  })
  listItem.appendChild(treatButton)
}

// Create a radios for voting
const voter = document.createElement('li')
voter.innerHTML = `${player.name}: `

// Check if player is alive
if (player.alive) {
  playerList.forEach(playerToVote => {
    // Ban players vote for themselves
    if (playerToVote.name === player.name) {
      return
    }
```

```
        const voteRadio = document.createElement('input')
        voteRadio.type = 'radio'
        voteRadio.name = `${player.name}`
        voteRadio.value = playerToVote.name

        const voteLabel = document.createElement('label')
        voteLabel.textContent = playerToVote.name
        voteLabel.htmlFor = voteRadio

        voter.appendChild(voteLabel)
        voter.appendChild(voteRadio)
      })
    }

    votesHtmlList.appendChild(voter)
  })
  if (voting) {
    votesHtmlList.style.display = 'block'
  } else {
    votesHtmlList.style.display = 'none'
  }
}

// Function to get player name from string
function cutString(str) {
```

```
  const colonIndex = str.indexOf(':')
  if (colonIndex !== -1) {
    return str.substring(0, colonIndex)
  } else {
    return str
  }
}

// Function to shuffle array
function shuffle(array) {
  for (let i = array.length - 1; i > 0; i--) {
    const j = Math.floor(Math.random() * (i + 1))
    const temp = array[i]
    array[i] = array[j]
    array[j] = temp
  }
  return array
}

// Function to give roles to all players
function giveRoles(playersList) {
  if (playersList.length < rolesList.length) {

    // Don`t start session if not enough players
    console.log('Not enough players to assign roles.')
    alert('Not enough players to assign roles.')
```

```
    return false

  } else {

    // Select roles for all players
    const players = shuffle(playersList)

    // Assign roles to players
    for (let i = 0; i < players.length; i++) {
      if (rolesList[i]) {
        players[i].selectRole(rolesList[i])
      }
    }
    updatePlayerList()
    return true
  }
}

// code

// Variables
let playerList = []
let voting = false
let gameStarted = false

// Get HTML elements
```

```
const playerHtmlList = document.getElementById('players')
const votesHtmlList = document.getElementById('votes')

const playerControlBox = document.getElementById('player-controls')
const voteControlBox = document.getElementById('vote-controls')

// Create HTML elements
const playerListElement = document.createElement('ul')
const createPlayerButton = document.createElement('button')
const createPlayerInput = document.createElement('input')

// Set attributes and text content
createPlayerButton.textContent = 'Create Player'
createPlayerButton.classList.add('button-3')
createPlayerButton.role = 'button'

createPlayerInput.placeholder = 'Enter player name'
createPlayerInput.id = 'create-player-input'

// Add event listener to create player button
createPlayerButton.addEventListener('click', () => {
  const playerName = createPlayerInput.value
  // Don`t create player if player name or role is empty
  if (!playerName) {
    alert('Please enter a player name.')
    return
```

```
  }

  // Don`t create new player if player already exists
  if (playerList.some(player => player.name === playerName)) {
    alert('Player already exists.')
    createPlayerInput.value = '' // Clear input field
    return
  }

  // Create player object and add to player list
  createPlayer(playerName, undefined)
  createPlayerInput.value = '' // Clear input field
  updatePlayerList()
})

// Create start game button
const startGameButton = document.createElement('button')
startGameButton.textContent = 'Start Game'
startGameButton.classList.add('button-3')
startGameButton.role = 'button'

// Add event listener to start game button
startGameButton.addEventListener('click', () => {
  const rolesAreGiven = giveRoles(playerList)
  if (rolesAreGiven) {
    gameStarted = true
```

```
    updatePlayerList()
    startGameButton.disabled = true
    createPlayerButton.disabled = true
    createPlayerInput.disabled = true
    dillerAndManiacButton.disabled = false
  } else {
    gameStarted = false
  }
})

// Create diller and maniac button
const dillerAndManiacButton = document.createElement('button')
dillerAndManiacButton.textContent = 'Create Diller and Maniac'
dillerAndManiacButton.classList.add('button-3')
dillerAndManiacButton.role = 'button'
dillerAndManiacButton.disabled = true

// Add event listener to d&m button
dillerAndManiacButton.addEventListener('click', () => {
  playerList[4].role = 'Diller'
  playerList[5].role = 'Maniac'
  updatePlayerList()
  dillerAndManiacButton.disabled = true
})

// Create vote button
```

```
const voteButton = document.createElement('button')
voteButton.textContent = 'Vote'
voteButton.classList.add('button-3')
voteButton.role = 'button'



// Add event listener to vote button
voteButton.addEventListener('click', () => {
  // Get all player votes
  let votes = []
  for (const voter of votesHtmlList.children) {
    try {
      const selectedPlayer = document.querySelector(`#votes input[name="${cutString(voter.textContent)}"]:checked`).value
      votes.push(selectedPlayer)
    } catch (error) {
      votes.push(null)
    }
  }

  // Sum player votes
  votes.forEach(vote => {
    const player = playerList.find(player => player.name === vote)
    if (player) {
      player.votes ++
    }
  })
```

```
  // Find player with most votes and kill him
  const playerWithMostVotes = playerList.reduce((playerWithMostVotes, player) => {
     return playerWithMostVotes.votes > player.votes ? playerWithMostVotes : player
   }, playerList[0])

  playerWithMostVotes.die()
  clearInterval(intervalId); // Stop the timer
  intervalId = null;
  startTime = null;
  timerButton.disabled = false; // Enable the button again
  timerDisplay.textContent = '00:00';
  voting = false
  updatePlayerList()
})

const timerButton = document.createElement('button')
timerButton.textContent = 'Start Timer'
timerButton.classList.add('button-3')
timerButton.role = 'button'

const timerDisplay = document.createElement('span')
timerDisplay.textContent = '00:00'
timerDisplay.id = 'timer'

let startTime = null;
```

```
let intervalId = null;

timerButton.addEventListener('click', () => {
  if (!startTime) {
    startTime = Date.now();
    intervalId = setInterval(updateTimer, 1000); // Update every second
    timerButton.disabled = true; // Disable button while timer is running
    voting = true
    updatePlayerList()
  }
});

function updateTimer() {
  const elapsedTime = Date.now() - startTime;
  const seconds = Math.floor(elapsedTime / 1000);
  const minutes = Math.floor(seconds / 60);

  // Check if the timer has reached 1 minute
  if (minutes >= 1) {
    clearInterval(intervalId); // Stop the timer
    intervalId = null;
    startTime = null;

    // Vote
    // Get all player votes
    let votes = []
```

```
    for (const voter of votesHtmlList.children) {
      try {
        const selectedPlayer = document.querySelector(`#votes input[name="${cutString(voter.textContent)}"]:checked`).value
        votes.push(selectedPlayer)
      } catch (error) {
        votes.push(null)
      }
    }
    // Sum player votes
    votes.forEach(vote => {
      const player = playerList.find(player => player.name === vote)
    if (player) {
      player.votes ++
    }
    })
    // Find player with most votes and kill him
    const playerWithMostVotes = playerList.reduce((playerWithMostVotes, player) => {
      return playerWithMostVotes.votes > player.votes ? playerWithMostVotes : player
    }, playerList[0])

    playerWithMostVotes.die()

    timerButton.disabled = false; // Enable the button again
    timerDisplay.textContent = '01:00';
    voting = false
```

```
    updatePlayerList()

    // Display an alert
    alert("Timer stopped at 01:00"); // Display an alert
  } else {
    // Continue updating the timer display
    const formattedTime = `${minutes.toString().padStart(2, '0')}:${(seconds % 60).toString().padStart(2, '0')}`;
    timerDisplay.textContent = formattedTime;
  }
}

// Append elements to the DOM
playerHtmlList.appendChild(playerListElement)
playerControlBox.appendChild(createPlayerInput)

playerControlBox.appendChild(createPlayerButton)
playerControlBox.appendChild(startGameButton)
playerControlBox.appendChild(dillerAndManiacButton)

voteControlBox.appendChild(voteButton)
voteControlBox.appendChild(timerButton)
voteControlBox.appendChild(timerDisplay)

// Initial update of the player list
updatePlayerList()
```

```
/* Buttons */


/* <button class="button-4" role="button">Button 4</button> */
.button-4 {
  appearance: none;
  background-color: #FAFBFC;
  border: 1px solid rgba(27, 31, 35, 0.15);
  border-radius: 6px;
  box-shadow: rgba(27, 31, 35, 0.04) 0 1px 0, rgba(255, 255, 255, 0.25) 0 1px 0 inset;
  box-sizing: border-box;
  color: #24292E;
  cursor: pointer;
  display: inline-block;
  font-family: -apple-system, system-ui, "Segoe UI", Helvetica, Arial, sans-serif, "Apple Color Emoji", "Segoe UI Emoji";
  font-size: 14px;
  font-weight: 500;
  line-height: 20px;
  list-style: none;
  padding: 6px 16px;
  position: relative;
  transition: background-color 0.2s cubic-bezier(0.3, 0, 0.5, 1);
  user-select: none;
  -webkit-user-select: none;
  touch-action: manipulation;
  vertical-align: middle;
```

```
  white-space: nowrap;
  word-wrap: break-word;
}

.button-4:hover {
  background-color: #F3F4F6;
  text-decoration: none;
  transition-duration: 0.1s;
}

.button-4:disabled {
  background-color: #FAFBFC;
  border-color: rgba(27, 31, 35, 0.15);
  color: #959DA5;
  cursor: default;
}

.button-4:active {
  background-color: #EDEFF2;
  box-shadow: rgba(225, 228, 232, 0.2) 0 1px 0 inset;
  transition: none 0s;
}

.button-4:focus {
  outline: 1px transparent;
}
```

```
.button-4:before {
  display: none;
}

.button-4:-webkit-details-marker {
  display: none;
}



/* <button class="button-3" role="button">Button 3</button> */
.button-3 {
  appearance: none;
  background-color: #a42e2e;
  border: 1px solid rgba(27, 31, 35, .15);
  border-radius: 6px;
  box-shadow: rgba(27, 31, 35, .1) 0 1px 0;
  box-sizing: border-box;
  color: #fff;
  cursor: pointer;
  display: inline-block;
  font-family: -apple-system,system-ui,"Segoe UI",Helvetica,Arial,sans-serif,"Apple Color Emoji","Segoe UI Emoji";
  font-size: 14px;
  font-weight: 600;
  line-height: 20px;
```

```
  padding: 6px 16px;
  position: relative;
  text-align: center;
  text-decoration: none;
  user-select: none;
  -webkit-user-select: none;
  touch-action: manipulation;
  vertical-align: middle;
  white-space: nowrap;
}

.button-3:focus:not(:focus-visible):not(.focus-visible) {
  box-shadow: none;
  outline: none;
}

.button-3:hover {
  background-color: #972c2c;
}

.button-3:focus {
  box-shadow: rgba(164, 46, 46, 0.4) 0 0 0 3px;
  outline: none;
}

.button-3:disabled {
```

```
  background-color: #d39494;
  border-color: rgba(27, 31, 35, .1);
  color: rgba(255, 255, 255, .8);
  cursor: default;
}

.button-3:active {
  background-color: #8e2929;
  box-shadow: rgba(70, 20, 20, 0.2) 0 1px 0 inset;
}


/* Input */
#create-player-input {
    padding: 5px 12px;
    font-size: 14px;
    line-height: 20px;
    color: #24292e;
    vertical-align: middle;
    background-color: #ffffff;
    background-repeat: no-repeat;
    background-position: right 8px center;
    border: 1px solid #e1e4e8;
    border-radius: 6px;
    outline: none;
    box-shadow: rgba(225, 228, 232, 0.2) 0px 1px 0px 0px inset;
```

```
    :focus{
        border-color: #d60303;
        outline: none;
        box-shadow: rgba(214, 3, 3, 0.3) 0px 0px 0px 3px;
    }
}



/* Page */
body {
    margin: 0;
    padding: 0;
    background-color: #2c2b31;
    display: flex;
    justify-content: center;
    align-items: center;
    flex-direction: column;
    color: #ffffff;
    font-family: "Freeman", sans-serif;
}

header {
    display: flex;
    height: 50px;
```

```
    justify-content: center;
    align-items: center;
    padding: 4px 0px;
}

header > img {
    height: 80%;
    margin-right: 4px;
}

header > span {
    color: #ffffff;
    font-family: "Freeman", sans-serif;
    font-weight: 400;
    font-size: 45px;
}

main {
    height: 90vh;
    width: 90vw;
    margin: auto auto;
    border-radius: 10px;
    background-color: #2c2b31;
    display: grid;
    grid-template-columns: 1fr 2fr;
    grid-template-rows: 1fr 1fr;
```

```
    justify-content: center;
}

main > * {
  background-color: #1f1f24;
  border-radius: 10px;
  margin: 2px;
}

#player-controls {
    display: flex;
    justify-content: center;
    align-items: center;
    flex-direction: column;
}

#player-controls > * {
    margin: 4px 0px
}

/* #players {
    overflow-y: scroll;
} */

#players > ul {
    display: grid;
```

```
    grid-template-columns: 1fr 1fr 1fr 1fr 1fr;
    grid-template-rows: auto;
    grid-gap: 4px;
    padding: 0;
    margin: 16px;
}

#players > ul > li {
    list-style-type: none;
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    border: #2c2b31 2px solid;
    border-radius: 5px;
    margin: 8px;
    padding: 10px;
}

#players > ul > li > img {
  width: 50px;
}

#votes {
  padding: 16px;
  /* overflow-y: scroll; */
```

```
}

#votes > ul {
  list-style-type: none;
}

#votes > li{
  list-style-type: none;
  margin: 2px 16px;
}

#votes > li > label {
  margin: 0 0 0 16px;
}

#vote-controls {
    display: flex;
    justify-content: center;
    align-items: center;
    flex-direction: column;
}

#vote-controls > * {
    margin: 4px 0px;
}
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

«ІНТЕЛЕКТУАЛЬНИЙ МОДУЛЬ ВИЗНАЧЕННЯ ЦІНИ НА НЕРУХОМІСТЬ»

Виконав: студент 4 курсу, групи 3КН-20б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Доценко І.К.
(прізвище та ініціали)

Керівник: к.т.н., професор каф. КН
Савчук Т. О.
(прізвище та ініціали)
«07» 06 2024 р.

Вінниця ВНТУ - 2024 рік

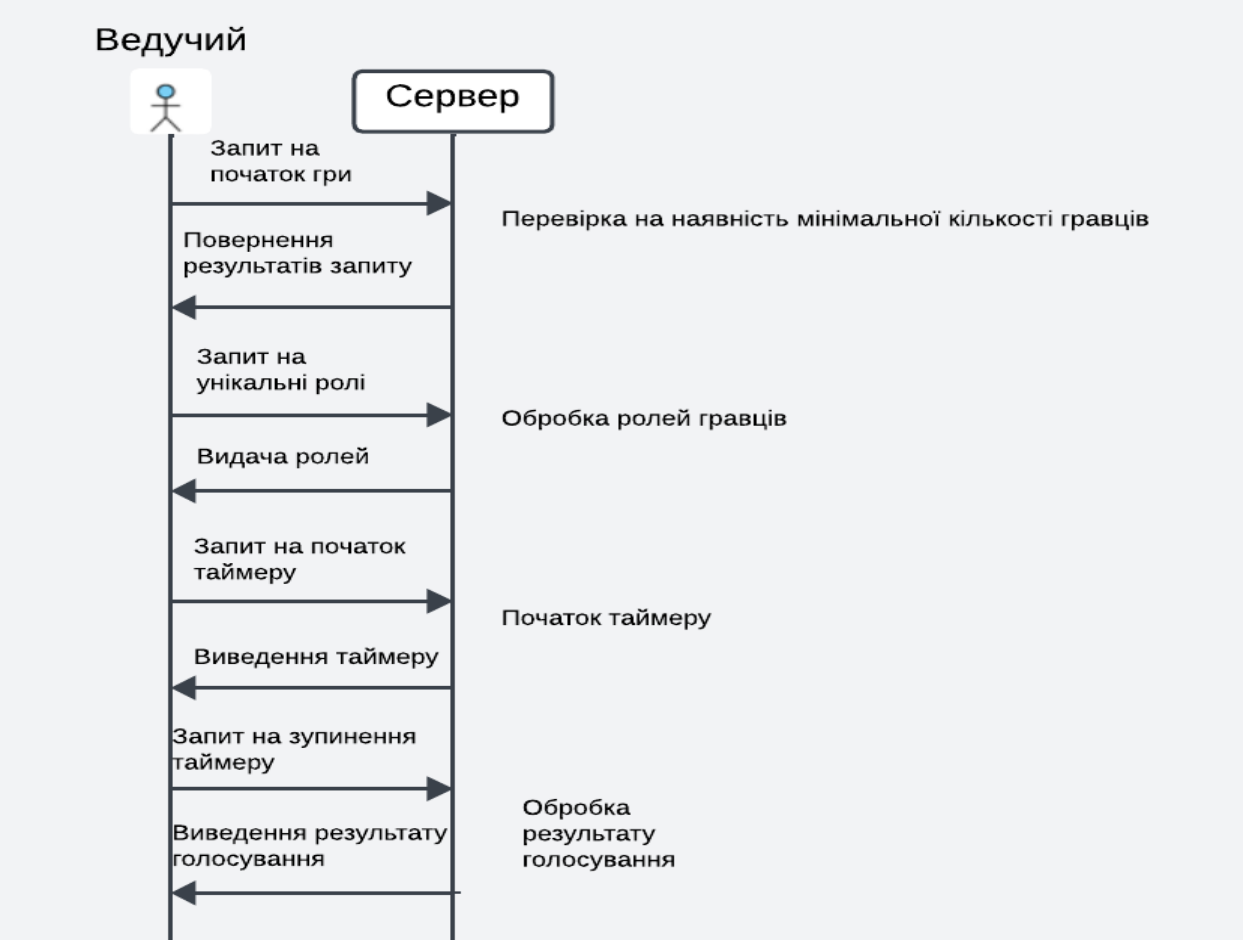


Рисунок 2.1 – UML-діаграма послідовності удосоконаленого алгоритму роботи гри “Мафія”

Рисунок 2.2 – UML-діаграма послідовності алгоритму вбивства та лікування

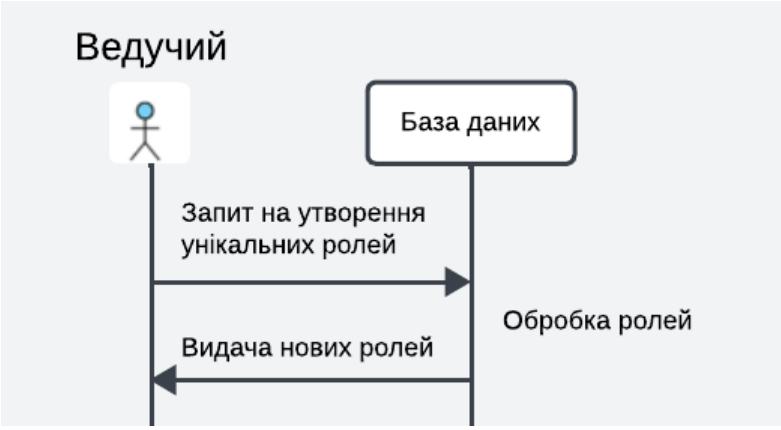


Рисунок 2.3 – UML-діаграма послідовності алгоритму отримання унікальних ролей

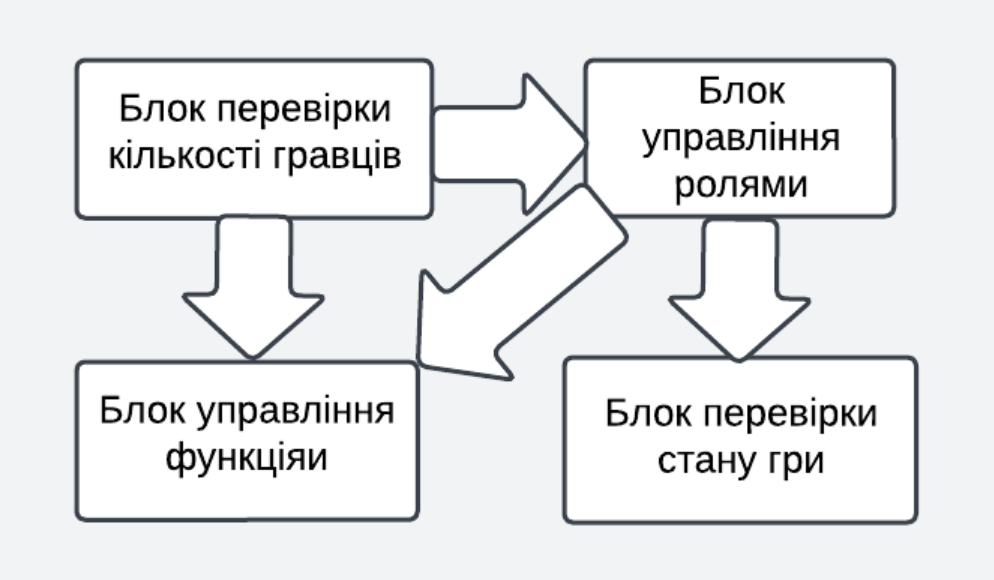


Рисунок 3.1 - UML-діаграма взаємодії блоків структури WEB-застосунку для ведучого гри “Мафія”