Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

Веб-сервіс для онлайн-гри у головоломку «L-game» на базі JS та Node.js
Назва теми

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр КвРІПЗ.200118.01.01.ПЗ

Виконав студент ІІІ курсу, група ІПЗс-20-1 _____ Підпис Я. М. Бадьора Ініціали, прізвище

Керівник к.т.н., доцент Науковий ступінь, звання _____ Підпис Ю. В. Форкун Ініціали, прізвище

Нормоконтролер канд. пед. наук, доцент Науковий ступінь, звання _____ Підпис Н. І. Праворська Ініціали, прізвище

До захисту допускаю:
Завідувач кафедри інженерії програмного забезпечення _____ Підпис Л. П. Бедратюк Ініціали, прізвище

5 червня 2023 р.

Хмельницький 2023

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій
Кафедра Інженерії програмного забезпечення
Рівень вищої освіти Перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ІПЗ
Л. П. Бедратюк
05.02 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Бадьорі Ярославу Михайловичу
Прізвище, ім'я, по батькові студента

1. Тема кваліфікаційної роботи Веб-сервіс для онлайн-гри у головоломку «L-game» на базі JS та Node.js

Керівник кваліфікаційної роботи Форкун Юрій Вікторович, к.т.н., доцент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 01.03.2023 р. №6
2. Строк подання студентом роботи на кафедру 05.06.2023 р.
3. Вихідні дані до роботи Матеріали переддипломної практики, методичні вказівки щодо його виконання для студентів спеціальності 121 «Інженерія програмного забезпечення»

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) Дослідження предметної області та постановка задачі, проектування програмного забезпечення, програмна реалізація та тестування.
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень) Три креслення (діаграма класів, діаграма варіантів використання, схема структури інтерфейсу користувача)

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Праворська Н. І., канд, пед, наук, доцент кафедри ІПЗ		
Антиплагіат	Гурман Іван Васильович, доцент кафедри ІПЗ	2.06.2023	3.06.2023

7. Дата видачі завдання « 05 » лютого 2023 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) Кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1 Ознайомлення з тематикою дипломного проектування, визначення та узгодження індивідуальних тем кваліфікаційних робіт (КвР)	01.12– 31.12.2022	
2 Збір матеріалу за темою КвР; дослідження предметної області, в якій планується використання програмного забезпечення (ПЗ), визначення задач та вимог, розробка технічного завдання	02.01 – 31.01.2023	
3 Проектування програмного забезпечення	01.02 – 28.02 2023	
4 Програмна реалізація з використанням відповідних засобів розробки	01.03 – 10.04.2023	
5 Тестування програмного забезпечення	11.04 – 30.04.2023	
6 Написання вступу, загальних висновків, оформлення переліку джерел посилання та додатків. Оформлення пояснювальної записки КвР згідно вимог	01.05 – 25.05.2023	
7 Попередній захист КвР	травень 2023 (згідно графіка)	
8 Перевірка КвР на плагіат, нормоконтроль, отримання відгуків, рецензій та інших супровідних документів. Брошурування (зшиття) пояснювальної записки.	26.05 – 30.05.2023	
9 Підготовка до захисту та захист КвР	з 01.06.2023	

Студент _____ Я. М. Бадьора
Підпис Ініціали, прізвище

Керівник роботи _____ Ю. В. Форкун
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема кваліфікаційної роботи «Веб-сервіс для онлайн-гри у головоломку «L-game» на базі JS та Node.js».

Автор роботи: Бадьора Ярослав Михайлович.

Керівник роботи: Форкун Юрій Вікторович.

Пояснювальна записка: 111 с., 7 рис., 3 табл., 5 дод., 40 джерел.

Графічна частина: 3 креслення.

ОНЛАЙН ГРА, JAVA SCRIPT, NODEJS, NOSQL, TYPESCRIPT.

Метою кваліфікаційної роботи є розробка веб-сервісу що дозволяє користувачам грати у головоломку L-game один проти одного в реальному часі, переглядати свої результати ти результати інших.

Під час кваліфікаційної роботи було проведено аналіз предметної області, виявлено проблематику існуючих засобів для гри у головоломку, проведено пошук і порівняння програмних рішень у розробці проектів подібного типу. Було запроектовано структуру проекту та успішно реалізовано проект веб-сервісу.

Для розробки веб-сервісу було використано JavaScript з бібліотеками Bootstrap та JQuery для клієнтської частини, та TypeScript на базі NodeJS з фреймворком NestJS для серверної частини.

В результаті було реалізовано веб-сервіс для онлайн-гри у головоломку «L-game» на базі JS та Node.js.

02.06.2023
Дата


Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	КвРІПЗ.200118.01.01.ПЗ	Пояснювальна записка	111		
2	А4		Завдання на кваліфіка-	1		
			ційну роботу			
3	А4		Анотація	1		
			Графічні документи			
4	А4		Презентаційні матеріали	15		

Змн.	Арк.	№ докум.	Підпис	Дата	КвРІПЗ.200118.01.01.ПЗ		
Виконав		Бадьора Я. М.		03.06	Веб-сервіс для онлайн-гри у головоломку «L-game» на базі JS та Node.js Відомість документів	Літ.	Арк. 1 / Аркушів 1
Керівник		Форкун Ю. В.		05.06			
Н. контр.				05.06		ХНУ, ІПЗс-20-1	
Зав. каф.		Бедратюк Л.П.		5.06			

ЗМІСТ

Змн.	Арк.	№ докум.	Підпис	Дата	КвРІПЗ.200118.01.01.ПЗ		
Виконав		Бадьора Я. М.		03.06	Веб-сервіс для онлайн-гри у головоломку «L-game» на базі JS та Node.js	Літ.	Арк. 5 / Аркушів 111
Керівник		Форкун Ю. В.		05.06			
Н. контр.		Праворська Н. І.		05.06		ХНУ, ІПЗс-20-1	
Зав. каф.		Бедратюк Л.П.		06.06			

ВСТУП

Останніми роками світ онлайн-ігор став свідком надзвичайного зростання, оскільки численні платформи пропонують широкий вибір ігор, щоб залучити та розважити гравців. Як частина цього простору, що розширюється, веб-головоломки набули значної популярності завдяки своїй доступності, інтелектуальним завданням і здатності об'єднувати гравців з різних куточків земної кулі. Цей дипломний проект спрямований на розробку веб-платформи, спеціально призначеної для L-ігри-головоломки.

L-гра, також відома як «L-puzzle», займає важливе місце в історії розважальної математики та логічних головоломок. Її винайшов Едвард де Боно, відомий психолог і автор, у 1969 році. Гра була розроблена для вивчення стратегічного мислення та здібностей до вирішення проблем, що робить її інтригуючим вибором для любителів головоломок.

Правила L-гри відносно прості, що додає їй привабливості для широкого кола гравців. Гра відбувається на прямокутній сітці, де кожен гравець по черзі кладе L-подібну плитку на дошку. Плитка складається з трьох або більше з'єднаних квадратів, які утворюють форму L. Мета полягає в тому, щоб розмістити плитки таким чином, щоб опонент не зміг зробити правильний хід. Гравець, який останній успішно розмістить плитку, виграє гру.

Загалом веб-головоломки набули надзвичайної актуальності в сучасну цифрову еру. Завдяки широкій доступності підключення до Інтернету та повсюдному поширенню веб-браузерів веб-ігри стали популярним вибором для розваг і розумової стимуляції. Зручність гри в головоломки безпосередньо з веб-браузера усуває потребу в установці спеціального програмного забезпечення, роблячи їх доступними для ширшої аудиторії У сфері веб-головоломок L-гра виділяється унікальним поєднанням простоти та стратегічної глибини. Він пропонує гравцям можливість покращити свої навички критичного мислення, планування та прийняття рішень. Розробляючи веб-платформу, присвячену L-грі, ми прагнемо надати ентузіастам зручне та захоплююче середовище для гри.

Метою цього дипломного проекту є розробка та реалізація веб-платформи з використанням NodeJS, NestJS, jQuery та Bootstrap, яка дозволяє користувачам реєструватися, входити в систему та брати участь у матчах L-ігор. Платформа полегшить не лише ігровий процес між незнайомцями, але й можливість запрошувати друзів, посилюючи соціальний аспект головоломки. Крім того, система відстежуватиме статистику гравців, дозволяючи їм контролювати свій прогрес і порівнювати свою продуктивність з іншими.

Щоб досягти поставленої мети потрібно виконати ряд завчасно сформованих завдань. По-перше, розробити інтуїтивно зрозумілий та зручний веб-інтерфейс з використанням Bootstrap та JQuery для забезпечення безперебійної навігації та оптимальної взаємодії з користувачем. По-друге, впровадити механізми автентифікації та авторизації користувачів за допомогою NodeJS і NestJS, що забезпечує безпечний доступ до платформи. По-п’яте, розробити надійну серверну систему для обробки даних користувачів, записів ігор і статистики, надання точної та актуальної інформації гравцям. Крім того, оптимізувати платформи для продуктивності, масштабованості та швидкості реагування, щоб пристосуватися до зростаючої бази користувачів і забезпечити плавний ігровий процес.

Розпочавши цей дипломний проект, ми прагнемо зробити свій внесок у світ веб-головоломок і надати ентузіастам цікаву платформу для перевірки своїх навичок стратегічного мислення за допомогою L-гри. Успішне завершення цього проекту не лише продемонструє технічну майстерність, але й стане цінним ресурсом для ентузіастів головоломок, які шукають онлайн-спільноту, щоб насолодитися цією захоплюючою грою.

1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Змістовний аналіз предметної області, її структурних та функціональних особливостей

Щоб розвинути всебічне розуміння предметної області ігрових веб-додатків, вкрай важливо провести аналіз, який охоплює різні аспекти. Цей аналіз дасть розуміння структурних і функціональних особливостей, необхідних для успішної реалізації веб-платформи для гри в головоломку L-ігри.

Почнемо з того, що аналіз предметної області ігрових веб-додатків розкриває динамічний ландшафт, який постійно розвивається. З розвитком технологій і зростаючим попитом на онлайн-ігри, ігрові веб-програми та додатки стають все більш поширеними. Ці програми пропонують користувачам зручність доступу та гри в ігри безпосередньо з веб-браузерів, усуваючи потребу в установці додаткового програмного забезпечення.

Розглядаючи інформаційні потреби користувачів у контексті веб-платформи L-ігор-головоломок, у центрі уваги потрапляють кілька ключових аспектів. Перш за все, користувачам потрібен бездоганний та інтуїтивно зрозумілий інтерфейс, який дозволяє їм легко переміщатися веб-сайтом. Це вимагає використання фреймворків, таких як jQuery та Bootstrap, які надають надійні інструменти для створення адаптивних і візуально привабливих веб-інтерфейсів.

Крім того, користувачі платформи головоломок L-game мають особливі інформаційні потреби, пов’язані з процесом гри та взаємодією з користувачем. Їм потрібен доступ до інформації, пов’язаної з грою, як-от правила L-гри, інструкції щодо гри та можливість переглядати ігрове поле під час гри. Також гравці будуть потребувати можливості збереження своїх результатів та порівняння їх з іншими гравцями та друзями.

Щоб задовольнити потенційний інтерес користувачів до окремих функцій, важливо проаналізувати конкретні функції, які, ймовірно, викликають найбільше залучення та задоволення. У випадку платформи L-game puzzle кілька функцій

мають значний потенціал для зацікавлення користувачів. Перш за все, можливість грати з незнайомими людьми є фундаментальною особливістю, яка дозволяє користувачам брати участь у складних матчах і відчувати гострі відчуття змагання. Ця функція дозволяє користувачам перевірити свої навички проти різноманітних супротивників, покращуючи загальний досвід гри.

Крім того, функція, яка дозволяє користувачам запрошувати друзів до гри, викликає значний інтерес у користувачів. Надаючи можливість спілкуватися з друзями та кидати їм виклик на матчі, платформа створює соціальний вимір, сприяючи дружній конкуренції та товариству між користувачами. Ця функція заохочує користувачів брати активну участь у роботі платформи, збільшуючи ймовірність тривалої взаємодії та утримання користувачів.

Крім того, функція перегляду статистики відіграє вирішальну роль у зацікавленні користувачів. Дозволяючи користувачам відстежувати свою продуктивність і порівнювати її з іншими, платформа створює відчуття досягнення та мотивацію для вдосконалення

Підсумовуючи, суттєвий аналіз предметної області ігрових веб-додатків, а також конкретних вимог та інтересів користувачів у контексті платформи L-game puzzle забезпечує міцну основу для розробки комплексного веб-рішення. . Ретельно враховуючи структурні та функціональні особливості, виявлені в результаті цього аналізу, ми можемо розробити та впровадити зручну та привабливу платформу, яка задовольняє інформаційні потреби користувачів, заохочує соціальну взаємодію та створює постійний інтерес і задоволення від гри L-game puzzle.

1.2. Аналіз наявного програмно-технічного забезпечення предметної області

Щоб розробити веб-платформу для гри в головоломку L-game, яка б відповідала сучасним потребам ринку програмних продуктів, важливо проаналізувати існуюче програмне забезпечення та технічну підтримку в

предметній галузі. Цей аналіз допоможе нам зрозуміти досвід провідних компаній-розробників програмного забезпечення та рішення, які вони впровадили в подібних проектах. Вивчаючи призначення, компанію-розробника, основні вікна інтерфейсу, переваги та недоліки цих аналогів, ми можемо приймати зважені рішення, щоб переконатися, що розроблене нами програмне забезпечення відповідає очікуванням ринку.

Першим було розглянуто Greenfoot.org — це освітня платформа, призначена для навчання концепціям програмування за допомогою Java та розробки візуальних ігор. Він забезпечує навчальне середовище для початківців для створення простих ігор, включаючи L-гру, використовуючи об’єктно-орієнтоване програмування. Компанія-розробник - Greenfoot Education Community створила його для Windows як основного інтерфейсу. Greenfoot.org включає інтегроване середовище розробки (IDE), де користувачі можуть писати Java-код, засіб перегляду проектів, ігрове полотно для тестування та запуску ігор, а також різні панелі для редагування об’єктів, функцій і класів.

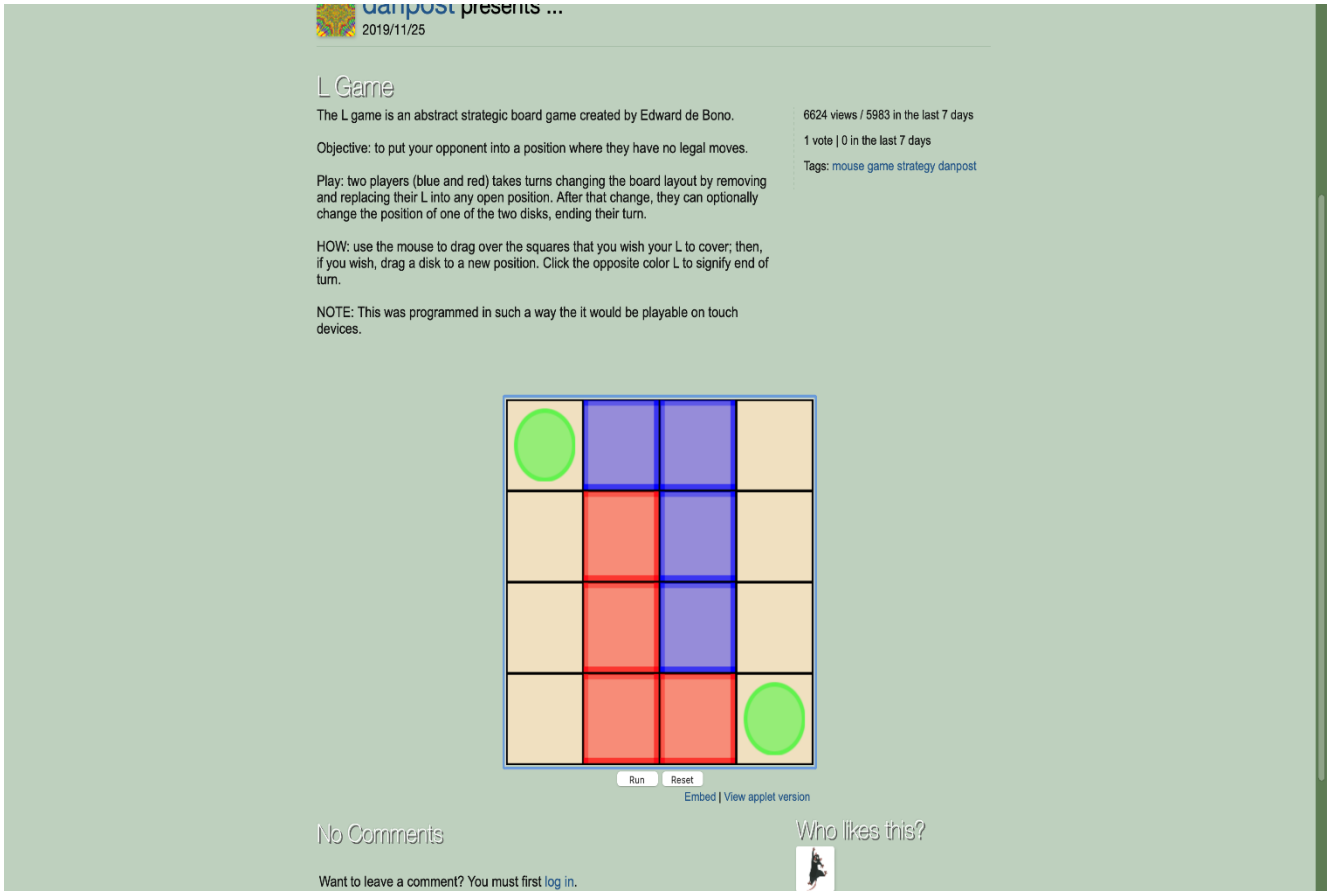


Рисунок 1.2.1 – Вигляд інтерфейсу Greenfoot.org

Переваги:

– орієнтований на освітні цілі та викладання концепцій програмування;
– забезпечує інтерактивне середовище для розробки простих ігор;
– пропонує широкий спектр освітніх ресурсів та навчальних посібників.

Недоліки:

– немає спеціального режиму для гри проти комп'ютера чи інших гравців;
– обмежена функціональність за межами освітніх цілей;
– застарілий дизайн та інтерфейс користувача.

Наступною було розглянуто L-гра від Vemodalen — це мобільний додаток, доступний на пристроях Android. Це дозволяє користувачам грати в L-гру проти суперника зі штучним інтелектом або інших гравців, надаючи цифрову адаптацію головоломки L-гри. Основний інтерфейс L-ігри від Vemodalen містить ігрову дошку, на якій відображаються L-плитки, область розміщення плиток супротивника, кнопки для виконання ходів і варіанти гри проти комп’ютера чи виклику інших гравців.

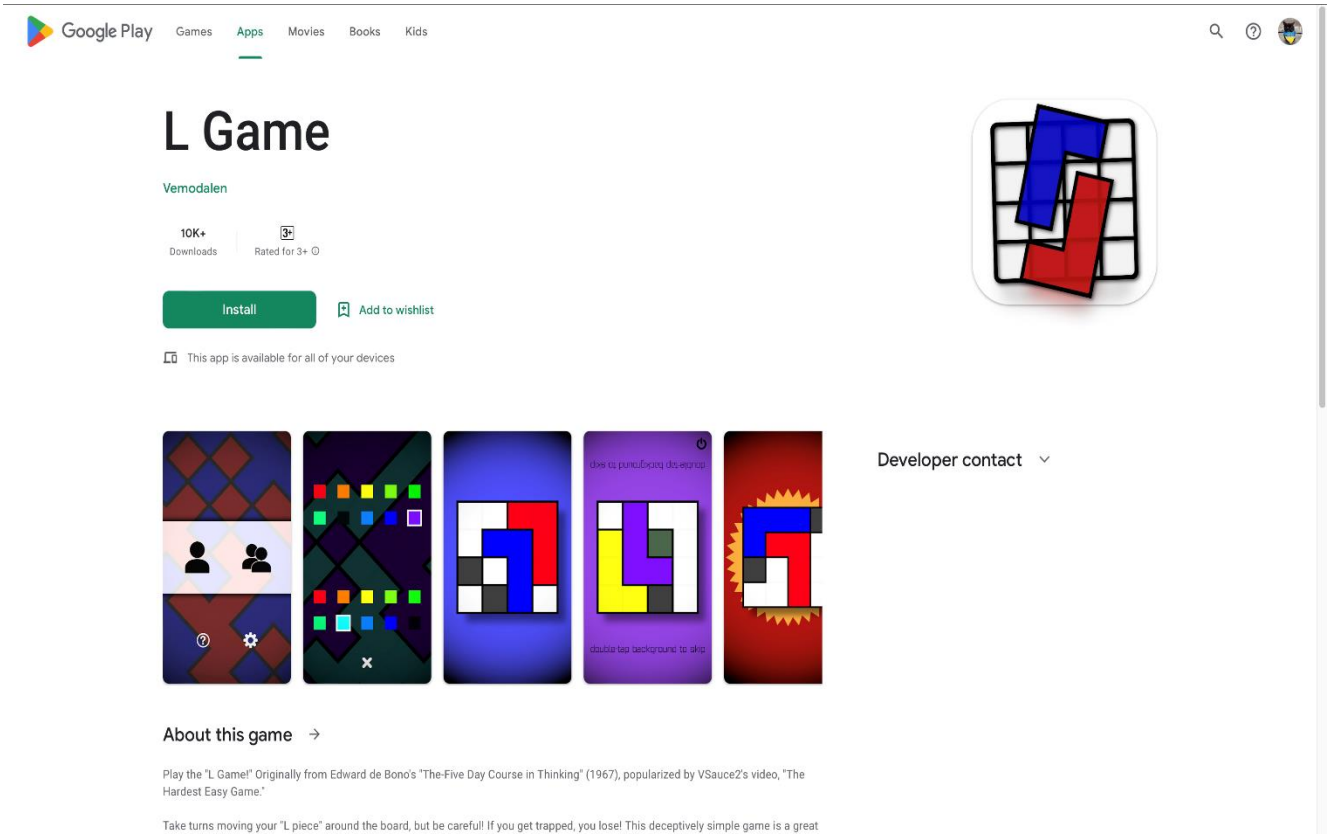


Рисунок 1.2.2 – Вигляд сторінки завантаженя Vemodalen

Переваги:

– сучасний дизайн і зручний інтерфейс;
– надає можливість грати проти AI або інших гравців.

Недоліки:

– обмежена підтримка платформи, доступна лише на пристроях Android;
– відсутність відстеження статистики для користувачів.

Наступною розглянуто веб-сайт hwwmath.looiwenli.com від компанії розробника Looi Wenli — це веб-платформа, яка пропонує режими «Гравець проти гравця» (PvP) і «Гравець проти оточення» (PvE) для гри в головоломку L-ігри. Він підтримує кросплатформну сумісність, що дозволяє користувачам грати з різних пристроїв на котрих підтримуються браузери.

Основний інтерфейс hwwmath.looiwenli.com містить ігрове поле для розміщення плиток, вікна чату для спілкування та кнопки для запрошення друзів або виклику випадкових супротивників.

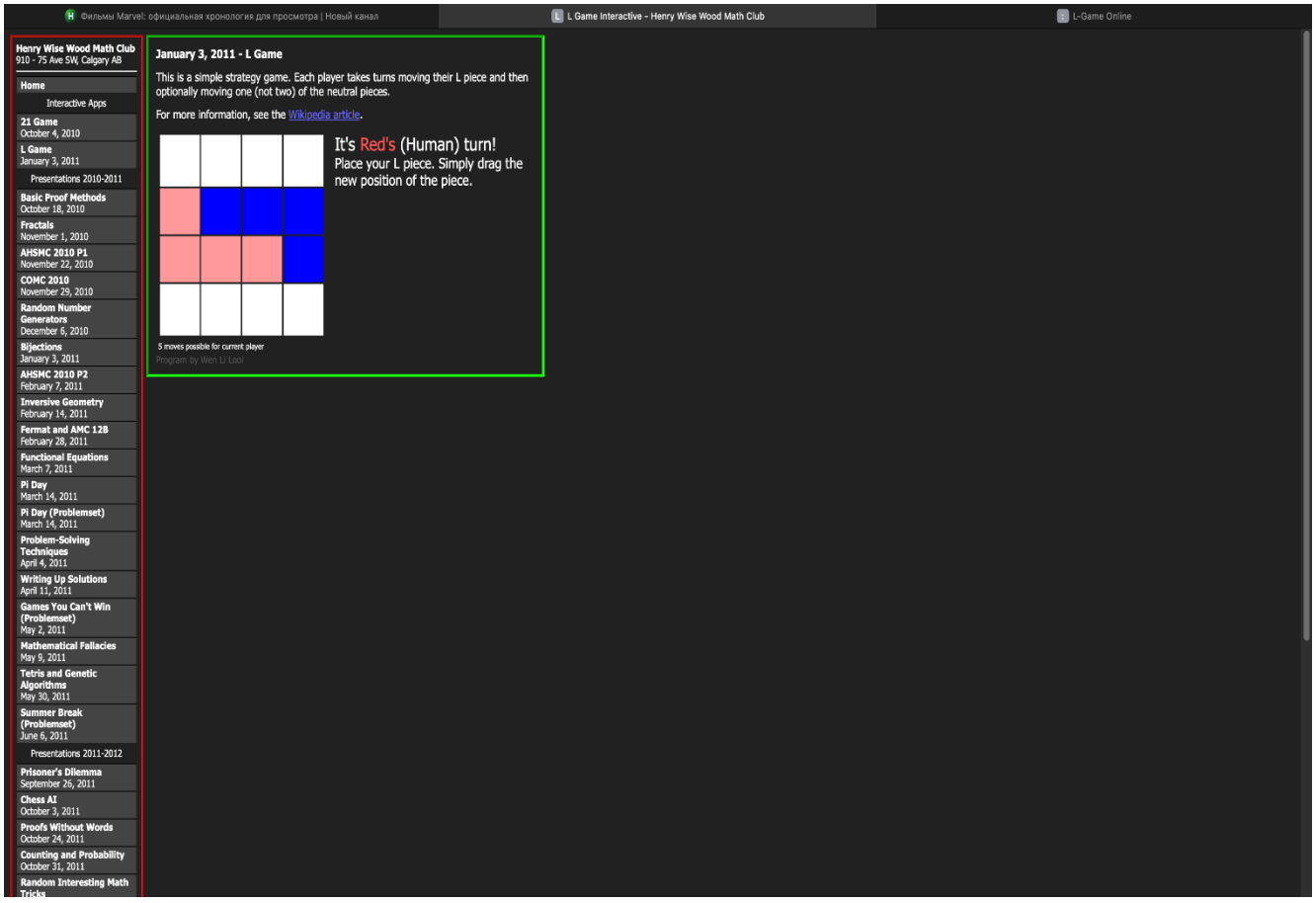


Рисунок 1.2.3 – Вигляд інтерфейсу hwwmath.looiwenli.com

Переваги:

– підтримує режими PvP і PvE, що задовольняють різні ігрові уподобання;

– кросплатформна сумісність, що дозволяє користувачам грати з різних пристроїв для гри.

Недоліки:

– застарілий дизайн та інтерфейс користувача;

– відсутність відстеження статистики для користувачів.

Порівняння цих існуючих застосунків наведено в додатку А Таблиця 1.

Аналізуючи ці існуючі програмні рішення, ми можемо визначити сильні та слабкі сторони кожного аналога.

Ці знання допоможуть нам легше приймати більш обґрунтовані рішення щодо вибору функцій, які будуть краще реалізовані на нашій веб-платформі L-game, гарантуючи, що вона відповідає сучасним функціональним потребам користувачів і подолає обмеження, які спостерігаються в аналогах. Крім того було отримано цінні знання про попит користувачів на ті чи інші функції котрі можна реалізувати у конкретному сервісі.

1.3. Аналіз вимог до програмного забезпечення

Щоб розробити веб-платформу для гри в головоломку L, було проведено детальний аналіз предметної області. На основі цього аналізу можна визначити та описати наступні вимоги до розробленого програмного забезпечення. Ці вимоги керуватимуть процесом розробки та допоможуть забезпечити відповідність програмного забезпечення потребам користувачів.

Інтерфейс проекту повинен бути простим і приємним. По-перше, простий інтерфейс покращує досвід користувача, зменшуючи когнітивне навантаження та роблячи навігацію інтуїтивно зрозумілою. Користувачі повинні мати можливість легко зрозуміти макет і функції веб-сайту, не стикаючись з плутаниною чи розчаруванням. Крім того, приємний інтерфейс сприяє задоволенню та залученню

користувачів, заохочуючи їх проводити більше часу на сайті. Це може призвести до збільшення утримання користувачів і покращення загального досвіду користувача.

Ігровий процес має бути спокійним і плавним, щоб забезпечити користувачам приємний і захоплюючий досвід. Щоб досягти цього, у грі повинні бути плавні та чутливі елементи керування, які дозволятимуть гравцям без зусиль пересувати частини головоломки або взаємодіяти з ігровим середовищем. Плавні переходи між різними станами гри, як-от початок нової гри або завершення рівня, сприяють плавному та безперервному ігровому процесу. Зведення до мінімуму відволікаючих факторів, таких як нав'язлива реклама чи надмірні візуальні ефекти, також може допомогти зберегти спокійну атмосферу під час гри.

Аналізуючи предметну область, можна виділити кілька функціональних можливостей для реалізації в проекті. До них належать:

– може бути передбачена система реєстрації користувачів, що дозволяє людям створювати облікові записи та безпечно входити в систему;

– доступ до гри повинен бути шквидким та прямим, тобто користувачі повинні витрачаи мінімум часу перед тим як почати гру;

– можливість користувачів грати з незнайомцями або запрошувати своїх друзів приєднатися до гри;

– може містити функцію для перегляду найкращих гравців за конкретний час, або за весь час;

– може містити функцію, яка дає змогу користувачам переглядати статистику гри, наприклад кількість зіграних ігор, співвідношення виграшів і програшів і середній час завершення;

– користувачі можуть мати можливість запрошувати своїх друзів грати в гру, надсилаючи прямі запрошення або через інтегровані платформи соціальних мереж;

– адаптація веб-сайту до різних розмірів екрана та пристроїв, що дозволить користувачам отримувати доступ і насолоджуватися грою на різних платформах, таких як настільні комп'ютери, ноутбуки, планшети та мобільні пристрої, без шкоди для ігрового процесу.

1.4. Постановка задачі

Після детеального аналізу предметної області та формування вимог до програмного забезпечення було поставлену задачу для роботи над проектом. В постановці задачі на роботу було окреслено наступні завдання:

– розробити мінімалістичний та приємний інтерфейс користувача, він має бути візуально привабливим, із добре розробленими макетами, колірними схемами та типографікою, а дизайн має бути інтуїтивно зрозумілим і зручним, щоб користувачі могли легко орієнтуватися в додатку та взаємодіяти з його різними функціями. Слід звернути увагу на загальну естетику та взаємодію з користувачем, щоб створити привабливе та приємне середовище для гри в головоломку L-game;

– розробити базу даних для збереження інформації про користувачів та заверешні матчі;

– розробити систему аутенфікації, тобто платформа повинна дозволяти користувачам створювати облікові записи, входити за допомогою своїх облікових даних і безпечно отримувати доступ до веб-сайту L-game puzzle;

– розробити гру з можливістю грати онлайн проти інших гравців в реальному часі, тобто користувачі повинні мати можливість грати проти незнайомців, приєднавшись до загальнодоступних ігрових кімнат або черги підбору гравців, а також користувачі повинні мати можливість запросити своїх друзів зіграти в головоломку L-ігри;

– розробити сервіс для перегляду статистики, тобто користувачі повинні мати доступ і переглядати свою історію ігор, включаючи кількість зіграних ігор, співвідношення перемог і поразок та іншу відповідну статистику;

– розробити систему рейтингу, тобто сервіс повинен включати систему оцінювання, яка дозволяє користувачам залишати відгуки та оцінювати свій досвід гри. Після кожної ігрової сесії у гравця збільшується або зменшується рейтинговий показник, що залежить від рейтингово показника опонента та вираховується за формулою.

1.5. Висновки з аналізу предметної області та постановки задачі

В цьому розділі було проведено ретельний аналіз предметної області та постановки проблеми. Аналіз зробив такі висновки:

Аналіз предметної області показав, що гра L-головоломка є стратегічною грою-головоломкою, в яку грають на квадратній сітці 5x5. Гра передбачає переміщення L-подібної фігури та пішаків по сітці, щоб перехитрити суперника. Це покрокова гра, де гравці роблять ходи по черзі. Ключові елементи гри включають L-подібну фігуру, пішаків і саму сітку. Аналіз також виявив потребу у взаємодії в режимі реального часу між гравцями та системою для обробки ігрової механіки, автентифікації користувачів і статистичного відстеження.

Постановка проблеми виявила відсутність доступної та зручної онлайн-платформи для гри в L-головоломку. Існуючі онлайн-платформи можуть не пропонувати комплексні функції, такі як керування обліковими записами користувачів, функціональні можливості для кількох гравців і комплексне відстеження статистики.

Було розроблено модель варіантів використання програми на основі зформованої задачі. Ця модель забезпечує повне розуміння поведінки та структури програмного забезпечення та можливі вузли використання, розуміння того як потрібно реалізовувати функціонал і складові програми, розділення і відповідність її до потреб користувача, зосереджуючись на тому, що має робити програмне забезпечення, щоб відповідати вимогам, визначеним під час аналізу. На основі аналізу постановка проблеми може бути визначена таким чином: розробити зручний веб-сайт для гри в L-головоломку, дозволяючи користувачам створювати облікові записи, входити в систему, грати з незнайомцями або запрошувати друзів до ігор і переглядати статистику. Веб-сайт повинен мати інтуїтивно зрозумілий і візуально привабливий інтерфейс користувача, одночасно забезпечуючи плавний ігровий процес і спілкування в реальному часі між гравцями.

2. ПРОЕКТУВАННЯ ВЕБ-СЕРВІСУ

2.1. Проектування системної структури

У розробці веб-платформи L-ігор-головоломок чітко визначена структура системи має вирішальне значення для організації різноманітних функцій і забезпечення ефективної взаємодії між різними підсистемами. У цьому підрозділі присвячено виділення програмних підсистем і способів взаємодії між ними.

Для даного продукту було обрано модульну структуру підсистем, у якій функціонал ділиться на незалежні окремі системи що взаємодіють з підлеглими модулями та між собою створюючи структуру вузлів і зв'язків між ними.

2.1.1. Підсистема користувача

Підсистема користувача відповідає за керування пов'язаними з користувачами функціями та взаємодією. Він обробляє реєстрацію користувачів, автентифікацію та керування профілями. Ця підсистема дозволяє користувачам входити в систему, запрошувати друзів грати, переглядати статистику та взаємодіяти з іншими користувачами на платформі. Крім того вона керує всіма засобами котрі допомагають користувачу ідентифікувати себе та інших між собою. Це підсистема контактує з всіма іншими підсистемами.

Функції:

– реєстрація користувача: дозволяє новим користувачам створювати облікові записи, надаючи необхідну інформацію та підтверджуючи облікові дані;

– автентифікація користувача: забезпечує безпечний механізм входу для користувачів до платформи за допомогою зареєстрованих облікових даних;

– керування профілем: дозволяє користувачам оновлювати інформацію профілю, керувати налаштуваннями та налаштовувати свої параметри;

– запрошення друзів: дозволяє користувачам надсилати запрошення до гри своїм друзям і ініціювати ігрові сеанси для кількох гравців;

– перегляд статистики: надає користувачам доступ до статистики їхнього ігрового процесу, включаючи дані про виграші та поразки, історію ігор, показники ефективності та коефіцієнт виграних матчів.

2.1.2. Підсистема матчів

Підсистема матчу обробляє ігрові функції головоломки L-ігри. Це полегшує пошук партнерів, ігрові сесії та ігрову логіку. Ця підсистема гарантує, що гравці можуть знаходити суперників, брати участь в ігрових сесіях і отримувати оновлення в реальному часі під час гри.

Функції:

– підбір гравців: полегшує процес пошуку відповідних суперників для гравців на основі таких факторів, як рівень навичок, доступні бажаний режим гри;

– керування ігровими сеансами: керує створенням, ініціалізацією та завершенням ігрових сеансів між гравцями;

– ігрова логіка: реалізує правила та логіку головоломки L-ігри, забезпечуючи чесний ігровий процес і стабільний ігровий процес;

– збереження результатів для ігрової статистики

– оновлення в реальному часі: дозволяє гравцям отримувати сповіщення та оновлення в реальному часі під час гри, включаючи ходи суперників і сповіщення про результати гри.

2.1.3. Підсистема авторизації

Підсистема авторизації гарантує, що користувачі мають належні права доступу та дозволи на платформі. Він керує ролями користувачів, контролем доступу та механізмами авторизації, збереженням данних про сесію користувача для підтримки безпеки та цілісності системи.

Функції:

– керування ролями: дозволяє адміністраторам визначати та призначати ролі користувачів із певними дозволами та привілеями;

– контроль доступу: встановлює обмеження доступу на основі ролей користувачів для захисту конфіденційних функцій і даних;

– механізми авторизації: реалізує протоколи автентифікації та авторизації для перевірки особи користувача та надання доступу до авторизованих ресурсів.

2.1.4. Способи взаємодії підсистем

Взаємодія підсистем користувача та матчу: підсистема користувача взаємодіє з підсистемою матчів, щоб ініціювати та керувати ігровими сеансами, обробляти запрошення та оновлювати статистику гравців; підсистема матчу взаємодіє з підсистемою користувача для автентифікації гравців, отримання профілів користувачів і оновлення інформації, пов'язаної з грою.

Взаємодія підсистем користувача та авторизації: підсистема користувача взаємодіє з підсистемою авторизації для перевірки облікових даних користувача, забезпечення контролю доступу та керування ролями користувачів.

2.1.5. Аналіз проектування структури підсистем

Розроблена структура системи, що зображена на Рисунку 2.1.5.1 гарантує, що підсистеми «Користувач», «Матч» і «Авторизація» безперебійно працюють разом, створюючи комплексну та інтерактивну веб-платформу головоломок L-ігор. Взаємодія між цими підсистемами дозволяє користувачам реєструватися, аутентифікуватися, запрошувати друзів, грати в ігри, переглядати статистику та забезпечує безпеку та цілісність системи за допомогою ролевих механізмів авторизації та контролю доступу.

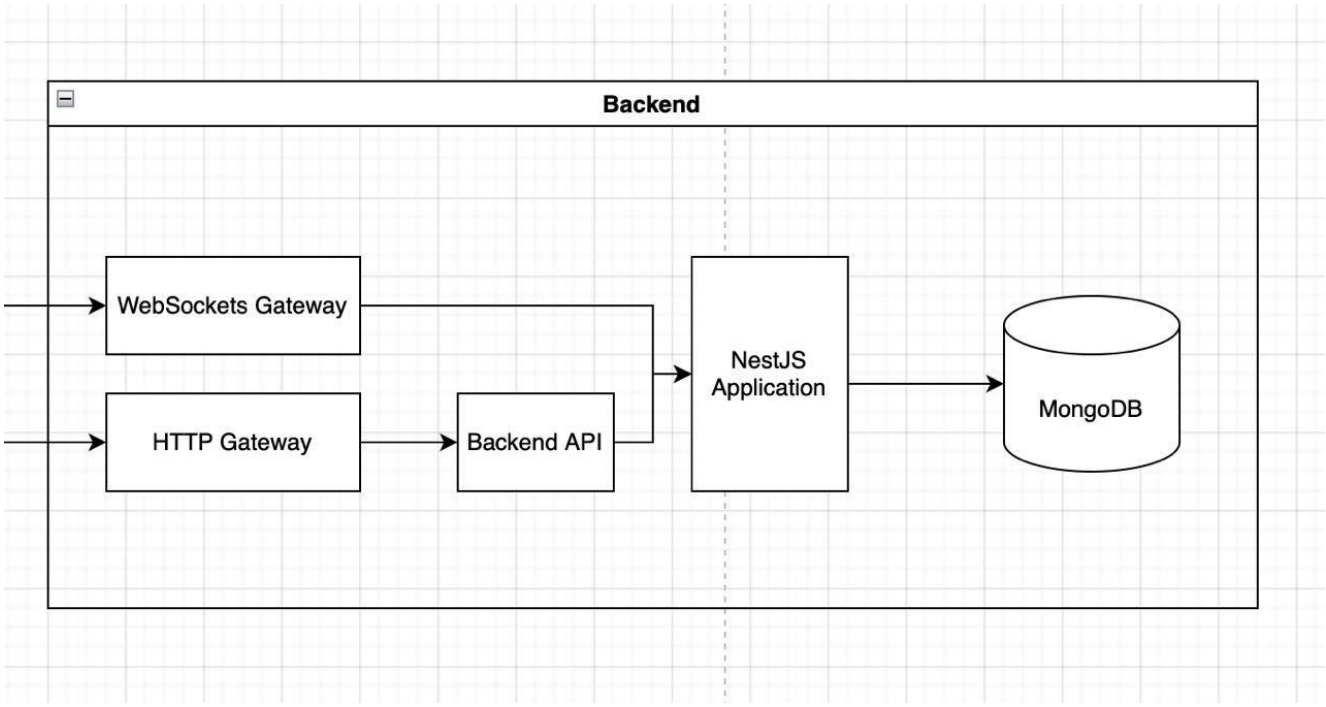


Рисунок 2.1.5.1 – Структура підсистем серверної частини та їх взаємодія

2.2. Проектування бази даних

Етап проектування бази даних є вирішальним кроком у розробці веб-платформи L-ігор-головоломок. У цьому підрозділі зосереджено увагу на важливості цього етапу проектування, надано вказівки щодо належного дизайну бази даних і коротко описано дизайн бази даних для головоломки L-ігри, включаючи дві основні таблиці: користувача та гри.

Етап проектування бази даних є важливим, оскільки він закладає основу для всієї програми. Добре спроектована база даних забезпечує ефективне зберігання, пошук і маніпулювання даними, що призводить до підвищення продуктивності, масштабованості та зручності обслуговування системи. Це також допомагає забезпечити цілісність даних, безпеку та дотримання бізнес-правил і вимог. Крім того проектування дозволяє створити ресурси які допоможуть команді розробки.

Щоб правильно спроектувати базу даних, важливо дотримуватися найкращих практик і враховувати конкретні потреби та вимоги програми.

Конструкція бази даних для платформи головоломок L-game включає дві основні таблиці: User і Game Match.

У таблиці «Користувач» зберігається інформація, пов'язана з користувачем, включаючи облікові дані та статистику. У таблиці Game Match відстежується інформація про кожен ігровий матч, включаючи гравців, які беруть участь, переможця, а також час початку та закінчення матчу. На Рисунку Б.2 зображена запроектована діаграма бази даних.

Завдяки належному дизайну та структуруванню бази даних вона забезпечує ефективне зберігання, пошук і керування даними про збіги користувачів і ігор, підтримуючи функціональність платформи L-game.

2.3. Проектування інтерфейсу

Дизайн інтерфейсу користувача (UI) є критично важливим аспектом дипломного проекту, який зосереджується на створенні інтуїтивно зрозумілого та привабливого веб-інтерфейсу для гри в головоломку L-game. У цьому розділі викладено особливості дизайну та елементи, необхідні для створення зручного та функціонального інтерфейсу користувача.

Простота є ключовим принципом дизайну інтерфейсу користувача веб-сайту L-game. Мета полягає в тому, щоб надати користувачам чистий і зрозумілий інтерфейс, який дозволить їм легко переміщатися сайтом, отримувати доступ до бажаних функцій і брати участь у ігровому процесі без зайвих складнощів. У дизайні слід віддавати перевагу чіткості, зручності використання, чистому та приємному візуальному вигляду.

Щоб забезпечити повну взаємодію з користувачем, такі елементи інтерфейсу необхідні для виконання ключових функцій на веб-сайті L-game puzzle:

Панель навігації: постійна панель навігації у верхній частині сторінки, яка забезпечує доступ до важливих розділів, таких як «Домашня сторінка», «Ігри», «Друзі», «Статистика» та «Акаунт».

Форма входу/реєстрації: форма, яка дозволяє користувачам увійти або зареєструвати новий обліковий запис, надаючи поля для даних користувача.

Грати з незнайомцями: кнопка або параметр, що дозволяє користувачам швидко приєднатися до гри з випадковими суперниками.

Запросити друзів: функція, яка дозволяє користувачам запрошувати своїх друзів зіграти в гру-головоломку L, вказавши посилання на гру.

Ігрова дошка: центральний елемент, на якому відображається L-подібна гра-головоломка, де користувачі можуть розміщувати та пересувати фішки під час гри.

Вікно чату: інтерактивне вікно чату, де користувачі можуть спілкуватися зі своїми суперниками або друзями під час гри.

Відображення статистики: спеціальний розділ або сторінка, де користувачі можуть переглядати статистику своєї гри, включаючи перемоги, поразки, рейтинги та іншу відповідну інформацію.

Сторінка профілю: сторінка профілю користувача, де користувачі можуть оновлювати дані свого облікового запису, змінювати зображення профілю та керувати своїми налаштуваннями.

Елементи інтерфейсу, згадані вище, мають взаємодіяти та взаємодіяти один з одним, щоб забезпечити цілісну взаємодію з користувачем. Панель навігації має бути однаковою на всіх сторінках, дозволяючи користувачам отримувати доступ до різних розділів веб-сайту з будь-якої сторінки. Форма для входу/реєстрації повинна надавати відповідні відгуки та повідомлення про помилки, щоб направляти користувачів через процес. Ігрова дошка має реагувати на дії користувача, як-от розміщення плиток або рух, і відображати оновлення в реальному часі під час гри. Вікно чату має сприяти обміну миттєвими повідомленнями між гравцями, дозволяючи їм спілкуватися та координувати рухи під час гри. Відображення статистики має оновлюватися динамічно залежно від ігрового процесу користувача, надаючи точну та актуальну інформацію.

Елементи інтерфейсу повинні бути розроблені таким чином, щоб вміщати різні параметри керування, забезпечуючи сумісність з різними пристроями та уподобаннями користувача. Це включає: адаптивний дизайн- інтерфейс повинен

адаптувати й оптимізувати свій макет і функціональність для різних розмірів екранів, таких як настільні ПК, планшети та мобільні пристрої; взаємодія дотику та миші - інтерфейс має підтримувати як взаємодію на основі дотику, що дозволяє користувачам взаємодіяти з ігровим полем та елементами інтерфейсу за допомогою жестів дотиком, так і традиційну взаємодію з мишею. Схему взаємодії цих елементів інтерфейсу зображено на Рисунку Б.2 .

Дизайн інтерфейсу користувача для веб-сайту головоломки L-game забезпечує інтуїтивно зрозумілий і приємний досвід для користувачів, підвищуючи взаємодію та задоволення, враховуючи простоту, включення основних елементів інтерфейсу, сприяння безперебійному зв'язку між елементами та розміщення різних варіантів керування.

2.4. Проектування ігрового модуля

Ігровий модуль є важливим компонентом дипломного проекту, який відповідає за полегшення функціональності ігрового процесу на веб-сайті L-game puzzle. Цей розділ присвячений аналізу ходу гри, формування взаємодії між користувачами та розробці структури ігрового модуля.

Для ефективної реалізації прогресу в грі важливо проаналізувати механіку та правила головоломки L-ігри. Розглянемо наступні аспекти:

– ініціалізація гри: створіть функцію для налаштування ігрового поля, ходів гравців і будь-яких необхідних початкових умов;

– ходи гравця: розвивайте логіку чергування ходів гравців, гарантуючи, що кожен гравець має можливість робити ходи в правильній послідовності;

– розміщення плиток: дозвольте гравцям розміщувати свої плитки на ігровій дошці, дотримуючись особливих правил головоломки L-ігри;

– переміщення плиток: увімкніть функції переміщення плиток, включаючи ковзання, обертання та гортання, відповідно до правил гри;

– перевірка ходів: запровадьте перевірку ходів, щоб переконатися, що рухи гравців відповідають правилам головоломки L-гри;

– умови виграшу: визначайте умови виграшу та програшу гравців.

Структура ігрового модуля повинна бути розроблена таким чином, щоб сприяти модульності, багаторазовому використанню та зручності обслуговування. Було визначення головні вузли ігрового модуля для проектування:

а) ініціалізація гри:
   – ініціалізація ігрової дошки;
   – налаштування черги гравця;
   – призначення початкові умови, наприклад початкові позиції плиток.

б) ходи гравця:
   – відстеження поточного гравця;
   – ходи по черзі.

в) розміщення плитки:
   – приймання даних гравця для розміщення тайлів на ігровому полі;
   – перевірка законності розміщення плитки;
   – оновлення ігровеого поля за допомогою розміщеної плитки.

г) рух плитки:
   – керування заданими діями руху плитки ти жетону, такими як ковзання, обертання та гортання;
   – підтвердження законності руху плитки;
   – оновлення ігрового поля за допомогою переміщеної плитки.

д) перевірка ходів:
   – впровадження правил та логіки для перевірки ходів гравців, реалізація функції пропуску ходу;
   – перевірка наявності незаконних рухів, таких як перекривання плиток або неправильне утворення L-форми.

е) умови виграшу:
   – постійне стеження за умовами виграшу на ігровому полі;

– визначення, коли гравець утворює L-подібну форму, і оголошування його переможцем;
– вирішення нічиї або нічиї, якщо це можливо.

Проводячи ретельний аналіз прогресу гри та розробляючи добре структурований ігровий модуль, веб-сайт L-game зможе забезпечити безперебійну та приємну гру для користувачів, дозволяючи їм грати з незнайомими людьми, запрошувати друзів і переглядати їхню статистику з легкістю та ефективністю.

2.5. Аналіз технологій і методів реалізації

2.5.1. Вибір платформи для реалізації

При розробці дипломного проекту зі створення сайту для гри в головоломку L-game вибір платформи для реалізації є критичним рішенням. При проектуванні веб сервісу було переглянуто найпопулярніші різновиди платформ.

Настільний комп'ютер - платформи що пропонують перевагу більшого розміру екрана та більшої обчислювальної потужності. Вони забезпечують більш багатий досвід користувача та дозволяють використовувати більш складні функції. Однак вони обмежені певними операційними системами та вимагають встановлення програмного забезпечення на пристрої користувача.

Мобільний – платформи що мають перевагу портативності, що дозволяє користувачам отримувати доступ до гри з будь-якого місця за допомогою своїх смартфонів або планшетів. Вони також надають такі функції, як сенсорна взаємодія та push-сповіщення. Однак мобільні платформи мають обмежений простір на екрані, що може вплинути на дизайн інтерфейсу користувача.

Браузер - платформа що є популярним вибором для веб-додатків, оскільки вона забезпечує сумісність між платформами, дозволяючи користувачам отримувати доступ до гри з будь-якого пристрою з веб-браузером. Це усуває потребу в розробці та встановленні для конкретного пристрою. Крім того, браузерні ігри можна легко оновлювати та розгортати, не вимагаючи від

користувачів завантажувати та встановлювати нові версії. Проте в кожної з платформ є свої специфічні недоліки, які теж були взяті до уваги.

Наприклад розробка для настільних платформ передбачає роботу з різними операційними системами (такими як Windows, macOS і Linux), кожна зі своїми вимогами та залежностями. Це може призвести до додаткових зусиль щодо розробки та обслуговування, а також до проблем із сумісністю.

Або ж розробка для мобільних платформ вимагає врахування різних розмірів екрана, роздільної здатності та можливостей пристрою. Це передбачає націлювання на кілька операційних систем (таких як iOS і Android) і дотримання відповідних інструкцій і політики магазину програм. Крім того, схвалення та оновлення магазину програм можуть призвести до затримок і додаткових проблем.

Крім того розробка для платформи браузера також має свої проблеми, такі як забезпечення крос-браузерної сумісності, обробка різних розмірів екрана та роздільної здатності та усунення вразливостей безпеки. Однак ці проблеми можна пом’якшити за допомогою стандартизованих веб-технологій і фреймворків.

Отже проаналізувавши зібрану інформацію, було вирішено що, платформа браузера є найкращим і найпростішим вибором для реалізації веб-сайту L-game.

По-перше, браузерні ігри пропонують чудову доступність, дозволяючи користувачам грати в гру на різноманітних пристроях, таких як настільні комп’ютери, ноутбуки, планшети та смартфони. Немає потреби в розробці або встановленні на конкретній платформі, що робить його зручним для користувачів.

По-друге, гра може досягти крос-платформної сумісності, використовуючи такі веб-технології, як HTML, CSS і JavaScript. Це гарантує безперебійну роботу гри в різних операційних системах і веб-браузерах, забезпечуючи широку доступність для користувачів незалежно від платформи для запуску сервісу, якій вони віддають перевагу.

Крім того, платформа браузера дозволяє легко оновлювати та розгортати гру. Оновлення можна безперешкодно розгортати, оскільки користувачі можуть отримати доступ до останньої версії безпосередньо через конкретний веб-браузер, що усуває потребу в установці вручну або проходженні складних процедур

затвердження в магазині програм. Більше того, користувачі зазвичай знайомі з веб-браузерами, що робить гру легкодоступною та інтуїтивно зрозумілою для навігації.

Знайомство з інтерфейсами перегляду дозволяє користувачам швидко адаптуватися до елементів керування та функцій гри.

Нарешті, розробка для платформи браузера забезпечує економічну ефективність. Замість того, щоб інвестувати в розробку для конкретної платформи, єдину кодову базу можна використовувати для кількох пристроїв і операційних систем. Такий підхід значно знижує витрати на розробку та обслуговування, що робить його більш економічним вибором.

Враховуючи ці фактори, платформа браузера виділяється як оптимальне рішення для реалізації веб-сервісу L-game, забезпечуючи широку доступність, звичність для користувачів, легке оновлення та економічну ефективність.

2.5.2. Вибір мови програмування для клієнтської частини

При аналізі мов програмування для клієнтської частини веб-сайту L-game розглядалися різні альтернативи JavaScript, зокрема Python

Розглянемо Python.

Переваги:

– має чіткий і стислий синтаксис, що робить його легким для розуміння;

– має велику спільноту розробників і широкий спектр бібліотек і фреймворків, що забезпечує швидкий розвиток;

– можна використовувати як для веб-розробки, так і для інших сфер, що робить його гнучким вибором.

Недоліки:

– переважно використовується для розробки на стороні сервера, і його підтримка браузером не така широка, як JavaScript;

– може бути повільнішим порівняно з JavaScript для певних операцій на стороні клієнта.

Проте JavaScript був обраний як мова програмування для клієнтської частини веб-сайту L-game puzzle з наступних причин:

– підтримується всіма основними веб-браузерами, що забезпечує широку доступність для користувачів;

– спеціально розроблено для веб-розробки, надаючи вбудовану функціональність і функції, призначені для створення інтерактивних веб-додатків;

– має величезну екосистему бібліотек, фреймворків і ресурсів, які сприяють швидкому розвитку та пропонують рішення для поширених проблем веб-розробки;

– забезпечує оновлення в реальному часі, інтерактивні користувацькі інтерфейси та динамічний вміст, забезпечуючи захоплюючий досвід гри без перезавантаження сторінок;

– має велике та активне співтовариство розробників, що полегшує пошук ресурсів, навчальних посібників та допомоги, коли це необхідно.

Загалом, сумісність JavaScript, веб-функції, широка екосистема та підтримка спільноти роблять його найкращим вибором для розробки клієнтської частини веб-сайту L-game puzzle.

2.5.3. Вибір мови програмування для серверної частини

При аналізі мов програмування для серверної частини веб-сайту L-game puzzle розглядалися альтернативи Node.js, зокрема Python, Java та C++.

Розглянемо Python.

Переваги:

– має чіткий, стислий та автоматично зрозумілий синтаксис, що робить його легким для читання та розуміння;

– має велику спільноту розробників і широкий спектр бібліотек і фреймворків, що забезпечує швидкий розвиток;

– можна використовувати для різних цілей, включаючи веб-розробку, і підтримує такі популярні веб-фреймворки, як Django і Flask.

Недоліки:

– GILPython може обмежувати продуктивність для багатопоточних програм;

– може працювати повільніше порівняно з іншими мовами, такими як Java або C++, для певних обчислювальних завдань.

Розглянемо Java.

Переваги:

– є популярною мовою програмування з великою та розвинутою спільнотою та розширеною документацією;

– програми Java можуть працювати на будь-якій платформі, яка підтримує віртуальну машину Java (JVM);

– має надійну підтримку паралелізму та масштабованості, що робить її придатною для обробки великомасштабних програм.

Недоліки:

– код Java, як правило, більш детальний порівняно з іншими мовами, що може призвести до більш тривалого часу розробки;

– має крутішу криву навчання порівняно з Node.js, особливо для початківців.

Розглянемо C++.

Переваги:

– відомий своєю ефективністю та високою продуктивністю, що робить його придатним для ресурсомістких програм;

– дозволяє керувати пам'яттю низького рівня та здійснювати прямий доступ до обладнання, забезпечуючи більший контроль над системними ресурсами.

Недоліки:

– є складною мовою, яка потребує глибокого розуміння керування пам’яттю та передових концепцій програмування;

– потрібно компілювати окремо для різних платформ, що робить його менш придатним для кросплатформної розробки.

Тому, Node.js було обрано як мову програмування для серверної частини веб-сайту L-game puzzle з наступних причин:

– оскільки клієнтська сторона веб-сайту вже розроблена за допомогою JavaScript, використання Node.js на стороні сервера забезпечує послідовність і дозволяє обмінюватися кодом і бібліотеками між клієнтом і сервером;

– використовує неблокуючі операції вводу-виводу та архітектуру, керовану подіями, що робить його масштабованим і ефективним для обробки кількох одночасних запитів;

– має яскраву та активну спільноту, яка пропонує широкий спектр пакетів і модулів через реєстр npm, що сприяє швидкій розробці та повторному використанню коду;

– забезпечує швидші цикли розробки завдяки своїй легкій і зручній структурі. Він підтримує швидке створення прототипів та ітераційну розробку;

– добре підходить для створення високомасштабованих програм завдяки своїй неблокуючій природі, що дозволяє ефективно обробляти велику кількість одночасних з'єднань.

Загалом, екосистема JavaScript, асинхронна природа, підтримка спільноти, швидкі цикли розробки та масштабованість роблять Node.js підходящим вибором для серверної частини веб-сайту L-game puzzle. Це забезпечує бездоганну інтеграцію з існуючим кодом клієнта JavaScript, сприяє ефективній обробці одночасних запитів і забезпечує надійне та масштабоване серверне рішення.

2.5.4. Вибір бази даних та СКБД

Розглянемо PostgreSQL:

Переваги:

– пропонує надійну підтримку реляційного моделювання даних і підтримує складні запити та транзакції;

– забезпечує цілісність і послідовність даних завдяки властивостям атомарності, узгодженості, ізоляції та довговічності (ACID);

– має активну спільноту, яка надає розширену документацію та підтримку.

Недоліки:

– налаштування та керування базою даних PostgreSQL може бути складнішим порівняно з базами даних NoSQL, такими як MongoDB;

– хоча PostgreSQL може обробляти великі набори даних і великий трафік, горизонтальне масштабування на кількох серверах може бути складним завданням.

Розглянемо MySQL:

Переваги:

– має велику базу користувачів і широку підтримку спільноти;

– відомий своїми швидкими операціями читання/запису та ефективним керуванням одночасними підключеннями;

– сумісний з різними операційними системами та може бути легко інтегрований з популярними веб-технологіями.

Недоліки:

– в основному розроблено для структурованих даних і не має вбудованої підтримки для зберігання складних і вкладених документів;

– горизонтальне масштабування MySQL може бути складним завданням, і для цього можуть знадобитися додаткові інструменти або конфігурації для розподілених налаштувань.

Отже MongoDB, база даних NoSQL, і MongoDB Atlas, хмарна служба баз даних, були обрані для веб-сайту L-game puzzle з таких причин:

– модель MongoDB на основі документів дозволяє зберігати складні та вкладені структури даних, що добре підходить для вимог гри;

– пропонує горизонтальну масштабованість шляхом шардингу даних на кількох серверах, що дозволяє веб-сайту справлятися зі зростаючими навантаженнями користувачів і зростаючими обсягами даних;

– архітектура та можливості індексування MongoDB забезпечують ефективні операції читання та запису, підтримуючи ігровий процес у реальному часі та інтерактивні функції;

– використовує гнучку мову запитів, яка підтримує розширені запити документів, що полегшує отримання та обробку даних;

– має велику та активну спільноту, яка надає ресурси, навчальні посібники та бібліотеки, керовані спільнотою, для допомоги в розробці.

MongoDB Atlas, як службу бази даних, було обрано з наступних причин:

– спрощує керування базами даних, включаючи надання, масштабування, резервне копіювання та моніторинг, зменшуючи операційні витрати;

– пропонує автоматичне горизонтальне масштабування, автоматичне резервне копіювання та конфігурації високої доступності, що забезпечує надійну продуктивність і довговічність даних;

– надає такі вбудовані функції безпеки, як шифрування в стані спокою, шифрування під час передачі та детальний контроль доступу, що забезпечує конфіденційність і захист даних;

– використовує хмарну інфраструктуру, забезпечуючи глобальне покриття, низьку затримку доступу та бездоганну інтеграцію з іншими хмарними службами.

Поєднання MongoDB як бази даних і MongoDB Atlas як служби керованої бази даних забезпечує гнучке, масштабоване, продуктивне та надійне рішення для зберігання даних для веб-сайту L-game. Він добре відповідає вимогам проекту та підтримує безперебійну розробку та розгортання веб-додатку.

2.5.5. Вибір фреймворків та бібліотек

NestJS серверний фреймворк, структуру Node.js, було обрано з таких причин:

– побудовано на TypeScript, який додає статичну типізацію та покращує якість коду, полегшуючи написання та підтримку масштабованих програм;

– використовує модульну та компонентну архітектуру одночасно з розширеним MVC патерном , що забезпечує кращу організацію коду, можливість повторного використання та обслуговування;

– забезпечує вбудовану підтримку ін’єкції залежностей, що спрощує керування залежностями та написання тестованого коду;

– підтримує горизонтальне масштабування та може обробляти великомасштабні додатки, що дуже важливо для правильного розміщення постійно зростаючої бази користувачів.

Mongoose, як бібліотеку моделювання даних було обрано з таких причин:

– спрощує роботу з MongoDB, надаючи підхід на основі схем для визначення моделей даних, забезпечуючи легку перевірку та структуровану обробку даних;

– пропонує багатий набір параметрів запитів і підтримує заповнення для ефективного отримання та роботи з пов'язаними даними;

– дозволяє визначати функції проміжного програмного забезпечення та хуки, забезпечуючи гнучкість для виконання спеціальної логіки до або після певних операцій бази даних;

– легко інтегрується з NestJS, забезпечуючи ефективні операції з базою даних у рамках NestJS.

Bootstrap, популярна інтерфейсна структура, була обрана з таких причин:

– пропонує адаптивні компоненти та утиліти для дизайну, що гарантує, що веб-сайт добре адаптується до різних розмірів екрана та пристроїв;

– надає широкий спектр попередньо розроблених компонентів інтерфейсу користувача та макетів, заощаджуючи час і зусилля на розробку;

– дозволяє налаштовувати відповідно до брендингу та вимог дизайну проекту, зберігаючи узгодженість на веб-сайті;

– велика спільнота розробників, що означає, що доступна обширна документація, ресурси та плагіни та теми, керовані спільнотою.

jQuery, як швидку та багатофункціональну бібліотеку JavaScript, було обрано з таких причин:

– спрощує роботу з DOM, полегшуючи взаємодію з елементами веб-сторінки та обробку дій користувача;

– надає інтуїтивно зрозумілий механізм обробки подій, що дозволяє розробникам легко прив'язувати події користувача та реагувати на них;

– пропонує вбудовану підтримку AJAX, що забезпечує безперебійний зв'язок із сервером і покращує інтерактивність веб-сайту;

– допомагає забезпечити узгоджену поведінку в різних браузерах, абстрагуючись від специфічних для браузера складнощів.

Вибравши NestJS як фреймворк, Mongoose для моделювання даних MongoDB, Bootstrap для зовнішнього дизайну та jQuery для покращеної інтерактивності на стороні клієнта, проект веб-сайту L-game отримує переваги від комплексного та інтегрованого стеку технологій. Ці варіанти дозволяють ефективну розробку, покращують взаємодію з користувачем і бездоганну інтеграцію між клієнтським і серверним компонентами веб-додатку.

2.5.6. Вибір засобів для розробки та інфраструктури

2.5.6.1. Вибір IDE

Visual Studio Code (VS Code) було обрано як IDE з кількох причин:

– це універсальна IDE із можливістю налаштування, яка підтримує широкий спектр мов програмування, включаючи JavaScript, TypeScript, HTML і CSS, які є ключовими для розробки веб-додатків, таких як веб-сайт L-game puzzle;

– доступний для операційних систем Windows, macOS і Linux, що гарантує, що розробники можуть працювати над проектом незалежно від платформи, якій вони віддають перевагу;

– незважаючи на багаті функції, VS Code легкий і оптимізований для продуктивності. Він має мінімалістичний інтерфейс, швидкий час запуску та низьке споживання ресурсів, що дозволяє розробникам безперебійно працювати навіть на менш потужних машинах;

– містить інтегрований термінал, що дозволяє розробникам запускати команди, виконувати налагодження та взаємодіяти з сервером проекту та базами даних, не виходячи з IDE;

– має величезну колекцію розширень, розроблених спільнотою, які пропонують додаткові функції та підтримку для різних інструментів, фреймворків

і мов. Ця екосистема підвищує продуктивність і полегшує розробку в різних сферах, таких як лінтування, форматування коду, контроль версій тощо;

– надає потужні функції IntelliSense, зокрема автозавершення коду, підказки щодо параметрів і спливаючі підказки документації;

– повністю інтегрується з Git, популярною системою контролю версій. Він надає візуальний інтерфейс для постановки, фіксації, розгалуження та злиття змін коду, спрощуючи процес співпраці для команд, які працюють над проектом;

– має велике й активне співтовариство розробників, що забезпечує доступ до документації, посібників, форумів і ресурсів. Ця мережа підтримки може бути корисною для вирішення проблем і пошуку вказівок під час процесу розробки.

Враховуючи універсальність, сумісність між платформами, легку вагу, інтегрований термінал, екосистему розширень, функції IntelliSense та навігації кодом, інтеграцію Git, а також активну спільноту, Visual Studio Code став ідеальним вибором для проекту. Він забезпечує зручне й ефективне середовище розробки, що дозволяє розробникам писати високоякісний код.

2.5.6.2. Вибір засобів комунікації

При розробці дипломного проекту «Веб-сайт для гри в головоломку L» було ретельно розглянуто вибір відповідних засобів комунікації для різних аспектів веб-сервісу.

Протокол HTTPS (Hypertext Transfer Protocol Secure) був обраний як засіб зв’язку для запиту та зберігання інформації у веб-сервісі.

Переваги HTTPS:

– шифрує дані, що передаються між клієнтом і сервером, забезпечуючи конфіденційність і цілісність обмінюваної інформації. Це вкрай важливо для захисту конфіденційних даних користувача, таких як облікові дані для входу та особиста інформація;

– використовує сертифікати SSL/TLS для автентифікації сервера, забезпечуючи клієнту впевненість у тому, що він взаємодіє з законною веб-службою, а не зі зловмисним об’єктом;

– широко визнаний і довіряє користувачам, сприяючи почуттю впевненості та надійності веб-сервісу;

Недоліки HTTPS:

– шифрування та дешифрування даних додає обчислювальних накладних витрат, що може дещо вплинути на продуктивність веб-служби;

– впровадження HTTPS вимагає отримання та налаштування сертифікатів SSL/TLS, що може бути складним завданням для новачків або тих, хто не знайомий із процесом;

– запити HTTPS зазвичай мають обмежені можливості кешування порівняно з HTTP, що потенційно впливає на ефективність доставки ресурсів.

Крім того, WebSockets були обрані як засіб зв'язку для синхронізації дій гравців у режимі реального часу. WebSockets забезпечують постійне з’єднання між клієнтом і сервером, забезпечуючи ефективний двонаправлений зв’язок. Рішення використовувати WebSockets було зумовлене такими факторами:

Переваги WebSockets:

– забезпечують миттєве оновлення та спілкування між гравцями під час гри. Це забезпечує більш інтерактивний і привабливий досвід користувача;

– за допомогою WebSockets немає потреби в частому опитуванні чи оновленні даних, що призводить до зменшення затримки мережі та кращого реагування;

– дозволяють як клієнту, так і серверу надсилати й отримувати дані, забезпечуючи бездоганну синхронізацію дій гравців.

Недоліки WebSockets:

– можуть не підтримуватися старими браузерами або певними мережевими конфігураціями;

– підтримка з’єднань WebSocket може споживати ресурси сервера, особливо коли мова йде про велику кількість одночасних користувачів. Для забезпечення

оптимальної продуктивності необхідні правильні стратегії управління та масштабування.

Загалом вибір протоколу HTTPS для пошуку та зберігання інформації забезпечує безпечний і надійний зв'язок, захищаючи конфіденційні дані користувача. Хоча й існують нелоіки такі як проблеми з кешуванням гет запитів, ненадійна структура формування запитів, довге шифрування ключів захисту і так діла. З іншого боку, WebSockets забезпечують синхронізацію дій гравців у реальному часі, покращуючи інтерактивний характер веб-сервісу. Незважаючи на те, що обидва підходи мають свої переваги та недоліки, рішення використовувати HTTPS і WebSockets було прийнято, щоб віддати перевагу безпеці, зв'язку в реальному часі та бездоганній взаємодії з користувачем на веб-сайті для гри в головоломку L-game.

2.5.6.3. Вибір засобів деплоя та хостингу

Під час аналізу доступних потенціальних сервісів для хостингу проекту були розглянуті такі популярні платформи хостингу, як Amazon Web Services (AWS), Microsoft Azure та DigitalOcean та Google CE. Кожна з цих альтернатив пропонує унікальні функції та можливості, які були ретельно вивчені з точки зору їхніх переваг і недоліків.

Розглянемо веб-сервіси Amazon.

Переваги AWS: надає широкий спектр послуг хостингу та має розвинену екосистему з широкою документацією та підтримкою спільноти. Він пропонує масштабованість, надійність і глобальну доступність.

Недоліки AWS: може бути складним у налаштуванні та управлінні, особливо для новачків. Цінова структура може бути складною, а додаткові послуги можуть призвести до додаткових витрат.

Розглянемо Microsoft Azure.

Переваги Azure: пропонує повний набір послуг хостингу, включаючи віртуальні машини, контейнери та безсерверні обчислення. Він добре інтегрується з іншими продуктами Microsoft, має чітку і перевірену структуру, має високу швидкодію і забезпечує чудову масштабованість.

Недоліки Azure: може мати крутішу криву навчання порівняно з іншими платформами, для розширених функцій знадобиться додаткова конфігурація.

Розглянемо DigitalOcean.

Переваги DigitalOcean: пропонує зручний інтерфейс і простий процес налаштування. Він забезпечує доступні тарифні плани, хорошу продуктивність і можливості масштабування.

Недоліки DigitalOcean: сервісні пропозиції відносно простіші порівняно з великими хмарними провайдерами, такими як AWS і Azure. Він може мати обмежену регіональну доступність порівняно з іншими платформами.

Після ретельного розгляду альтернатив, Google Compute Engine (GCE) було обрано як систему хостингу для веб-сайту L-game. На це рішення вплинули наступні фактори:

– забезпечує чудову масштабованість, дозволяючи веб-сайту обробляти різні рівні трафіку користувачів. Він пропонує гнучкі ресурси, які можна легко збільшити або зменшити залежно від попиту. Ця масштабованість забезпечує оптимальну продуктивність і швидкість реагування для користувачів;

– Google має міцну репутацію в якості надійної інфраструктури. Основна архітектура Compute Engine розроблена для забезпечення тривалого часу безвідмовної роботи та мінімізації перерв у роботі. Ця надійність має вирішальне значення для підтримки безперебійної взаємодії з користувачем;

– розгалужена глобальна мережева інфраструктура Google забезпечує підключення з низькою затримкою та швидку передачу даних. Оскільки центри обробки даних стратегічно та вдало розташовані по всьому світу, користувачі з різних географічних регіонів можуть отримати постійний доступ до веб-сайту L-game з мінімальною затримкою;

– забезпечує надійні функції безпеки, зокрема вбудовані правила брандмауера, шифрування даних і керування ідентифікацією. Він дотримується галузевих стандартів безпеки, забезпечуючи захист даних користувачів і відповідність нормам;

– за допомогою Google Compute Engine проект може легко інтегруватися з іншими хмарними службами Google. Ця інтеграція забезпечує ефективне зберігання, керування базами даних і доставку вмісту, підвищуючи загальну функціональність і продуктивність веб-додатку;

– пропонує багатий набір інструментів розробника, API та вичерпну документацію для керування та розгортання програм на Compute Engine. Крім того, Google надає відмінну технічну підтримку, забезпечуючи оперативну допомогу та керівництво протягом життєвого циклу проекту.

Враховуючи масштабованість, продуктивність, надійність, глобальну мережеву інфраструктуру, функції безпеки та відповідності вимогам, інтеграцію з хмарними службами Google, а також інструменти розробника та підтримку, які пропонує Google Compute Engine, він став оптимальним вибором для розгортання та розміщення проекту. Використовуючи Google Compute Engine, проект може отримати переваги від надійної, масштабованої та безпечної інфраструктури хостингу, що забезпечує безперебійну роботу користувача та ефективне керування веб-програмою.

2.6. Результати проектування веб-сервісу

Таким чином, веб-сервіс для гри в головоломку L-game був запроектований згідно поставлених завдань із такими ключовими функціями:

– багатокористувацька функція, що дозволяє користувачам грати з незнайомцями та запрошувати друзів приєднатися до гри;

– система зв'язку в режимі реального часу для безперебійної взаємодії між гравцями під час гри;

– відстеження статистики користувачів і таблиці лідерів для демонстрації найефективніших гравців;

– чуйний та інтуїтивно зрозумілий інтерфейс користувача для візуально привабливого та зручного використання;

– MongoDB як система керування базами даних для ефективного зберігання даних у зручному форматі;

– фреймворк NestJS і TypeScript для розробки на стороні сервера, що забезпечує модульну та масштабовану структуру;

– розгортання в системі хостингу Google Compute Engine для надійного та глобального доступу.

Підсумовуючи, фаза проектування була завершена охоплюючи всі необхідні компоненти, щоб забезпечити привабливий та інтерактивний досвід для користувачів. Дизайн забезпечує ефективність, безпеку та масштабованість, що робить веб-сайт L-game перспективним проектом для реалізації.

Взаємодія з користувачем веб-сайту була ретельно розроблена, щоб забезпечити бездоганний і приємний інтерфейс. Користувачі можуть легко орієнтуватися в інтуїтивно зрозумілому та візуально привабливому дизайні, що дозволяє їм входити в систему, грати з незнайомими людьми або запрошувати друзів, а також без зусиль переглядати статистику ігрового процесу.

Архітектура веб-сервісу складається з трьох основних модулів: клієнтського модуля, серверного модуля та модуля бази даних. Ці модулі працюють разом, щоб полегшити спілкування та обмін даними. Кожен модуль містить підмодулі, які ефективно вирішують конкретні завдання, забезпечуючи безперебійну роботу.

Технологічно веб-сайт використовує HTML, CSS і JavaScript для інтерфейсної розробки, забезпечуючи сумісність у різних веб-переглядачах. На серверній частині Python і Flask використовуються для обробки на стороні сервера, що забезпечує ефективну роботу.

Для забезпечення безпечного доступу веб-сайт використовує надійну систему автентифікації користувачів. Користувачі можуть створювати облікові записи та безпечно входити, використовуючи свої облікові дані. Паролі

зашифровані, щоб захистити інформацію користувача від несанкціонованого доступу зловмисників.

Багатокористувацька функція дозволяє користувачам брати участь в іграх з незнайомцями або запрошувати друзів. Спілкування в режимі реального часу забезпечується через Websockets, що забезпечує безперебійну взаємодію та синхронізацію рухів між гравцями.

Користувачі також мають можливість відстежувати статистику свого ігрового процесу, включаючи кількість зіграних ігор, перемоги, поразки та інші відповідні показники. Ця функція додає елемент змагання та спонукає користувачів вдосконалювати свої навички.

Ефективність, безпека та масштабованість є найважливішими міркуваннями при проектуванні. Веб-сайт розроблено для роботи з кількома одночасними користувачами, що забезпечує швидкий час відповіді. Заходи безпеки, такі як впровадження протоколу HTTPS для безпечної передачі даних, захищають інформацію користувача.

Підсумовуючи, на етапі розробки проекту «Веб-сайт для гри в L-головоломку» успішно розглянуто всі необхідні компоненти. Вибрані технології, архітектура та функції забезпечують ефективність, безпеку та масштабованість. Наступні кроки включають етап впровадження та тестування, на якому розроблені концепції будуть перетворені на повнофункціональний веб-сайт.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1. Модулі веб-сервісу та способи комунікації між ними

Після завершення загального дизайну структури веб-сервісу та проведення всебічного аналізу для вибору відповідних технологій для впровадження фокус змістився на детальне проектування різних модулів програми. Це включало розбиття системи на більш дрібні компоненти, визначення їх функціональних можливостей і окреслення стратегій їх реалізації.

Окрім дизайну модуля, увагу було приділено розробці інтерфейсу користувача, що забезпечує інтуїтивно зрозумілий та візуально привабливий дизайн, який покращує взаємодію з користувачем. Інтерфейс був ретельно розроблений, щоб забезпечити можливість входу в систему, гри з незнайомими людьми, запрошення друзів і безперешкодного перегляду статистики користувачів.

Одночасно розпочався етап впровадження, на якому вибрані технології, були використані для втілення розроблених концепцій у життя. Це передбачало написання необхідного коду, створення необхідної функціональності на стороні сервера та клієнта та інтеграцію системи керування базами даних для ефективного зберігання та пошуку даних.

Крім того, були сформульовані рекомендації щодо роботи та використання веб-сервісу, які містять чіткі інструкції щодо навігації програмою, використання її функцій та взаємодії з іншими користувачами. Ці інструкції спрямовані безпосередньо на забезпечення безперебійної роботи інтерфейсу користувача та сприяння залученню нових користувачів.

Для забезпечення якості та надійності розробленого веб-сервісу була визначена комплексна методологія тестування. Це передбачало планування та виконання різних типів тестування, таких як модульне тестування, інтеграційне тестування та тестування прийнятності користувача. Процес тестування був спрямований на виявлення та вирішення будь-яких потенційних помилок, помилок або проблем із продуктивністю, забезпечуючи надійний і стабільний продукт.

Цей розділ присвячений етапу впровадження та тестування програмного забезпечення дипломного проекту. Він охоплює детальний дизайн модулів додатків, впровадження інтерфейсу користувача, управління даними, робочі вказівки та методологію ретельного тестування.

В основі проекту лежать три модулі найвищого рівня, які поділяють веб-сервіс на окремі частини. Кожна з цих частин відповідає за свою сферу сервісу, наприклад клієнтський модуль обслуговує роботу частини програми що відбувається на стороні клієнту що завантажується на комп'ютер користувача в процесі роботи з програмою. Такий розділ зумовлений тим що деякий програмний код потрібно виконувати окремо від основного Ці модулі включають в себе під-модулі, або ж логічні модулі, що мають різну форму в залежності від способу та структури реалізації.

3.1.1. Клієнтський модуль

Загалом клієнтський модуль відповідає за інтерфейс користувача, щоб дати йому можливість користуватися всіма можливостями веб-сервісу: авторизація, перегляд статистики та гру в L-game. Крім того цей модуль реалізує саму гру, її логіку та комунікацію за допомогою сервера між двома гравцями.

Під час детального проектування було визначено кілька частин які повинен реалізувати клієнтський модуль: міжсторінкова навігація, сервіс для взаємодії з API серверу та авторизації запитів, імплементація гри, імплементація комунікації з клієнтом опонента по грі.

Реалізувати клієнтський модуль було вирішено за допомогою принципу рендерингу на стороні серверу, за яким сторінка формується спочатку на сервері за шаблоном, потім до неї підключаються скрипти та стилі що також зберігаються на сервері, після чого сторінка готова до відображення.

Для такої реалізації було використано нативну бібліотеку серверного фреймворку NestJS – nestjs/common. Ця бібліотека підтримує різні варіації

шаблонів для html розмітки. В нашому випадку було обрано EJS як найпопулярніший та найзручніший.

Шаблони зберігаються в окремій папці, де кожен файл відображає шаблон конкретної сторінки:

```
views/
      login.ejs
      menu.ejs
      register.ejs
      room.ejs
      stats.ejs
```

Ці шаблони є звичайними файлами html розмітки, в які, за допомогою спеціальних конструкцій, сервер може додавати дані:

```
<div id="game">
      <div id="game-id"><%= roomId %></div>
</div>
```

Для реалізації більшості динамічних функцій клієнтської частини використовуються JS скрипти, що підключаються до сторінки звичайним шляхом – за допомогою розділу <script></script>:

```
<script src="/js/api.js"></script>
```

Для підключення зовнішніх бібліотек, тобто тих скриптів які зберігаються не на домашньому сервері додатку, використовуються додаткові теги:

```
<script
      src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.b
undle.min.js"
></script>
```

Схожа методика використовується й для CSS стилів:

```
<link rel="stylesheet" href="/room.css">
<link
     href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap
.min.css"
     rel="stylesheet"
     ...
/>
```

Внутрішні скрипти було вирішено реалізувати у форматі JS модулів в окремих файлах, які можуть окремо бути підключені до сторінок в разі необхідності. Кожен такий модуль відповідає за свою сферу відповідальності:

```
public/js/
     game/
          game.js
          game-state.js
          tiles-container.js
          ...
     helpers/
          creds.js
          ...
     api.js
     login.js
     menu.js
     stats.js
     ...
```

Кожен скрипт як правило містить один або кілька споріднених класів і/або інструкції для виконання при підключені скрипту:

```
class Game {
```

```
    elem = null
    tilesContainer = null

    // ...

    constructor(gameElem) {
        // ...
    }

    init() {
        // ...
    }

    // ...

}

const main = async () => {
    // ...
    const gameElem = $('#game')
    const game = new Game(gameElem)
    game.init()
}
$(() => { main().catch(e => { throw e }) })
```

Навігацію між сторінками реалізовано як і звичайними посиланнями так і засобами EventHandlers з JQuery, що дозволяють нам підписуватись на події зв'язані з натисканням і виконувати потрібні інструкції:

```
$('#menu-connect-button').on('click', () => {
    (async () => {
        const match = await LGameAPI.findMatch()
        if(!match) {
            document.location.href = '/play'
```

```
        } else {
            document.location.href = /room/${match.socketRoomId}
        }
    })()
});
```

Відправка запитів до серверу реалізована за допомогою технології AJAX яку поставляє пакет JQuery. Для комунікації з API серверу було створено окремий JS модуль, що включає статичний клас котрий викликає відповідні ендпоінти і обробляє полики зв'язані з комунікацією з сервером. Крім того цей модуль відповідає за авторизацію запитів, та працює з cookies веб сервісу, щоб зберігати та відслідковувати токен для авторизації.

Реалізація гри відтворена за допомогою розбиття на окремі JS модулі (див. Рисунок 2.4.1) кожен з яких упорядковується головному JS модулю Game, котрий займається управлінням над синхронізацією гри з сервером і між гравцями та контролем підлеглих модулів.

3.1.2. Серверний модуль

Серверний модуль в цілому призначений для того щоб надавати та зберігати інформацію про користувачів та про зіграні матчі між користувачами. Крім того цей модуль відповідає за синхронізацію і з'єднання між опонентами у грі.

На етапі проектування було вирішено реалізувати серверну частину за допомогою NodeJS та фреймворку NestJS що використовує TypeScript як мову скриптингу і має структуру зібрану з модулів за принципом DI.

Структура цього модуля сформована за принципом Dependency Injection, де підмодулі об'єднуються в дерево за допомогою зв'язків між собою. Цей підхід дозволяє строго структурувати відносини між різними частинами проекту, полегщити менеджемент звязків мін ними, покращує масштабованість і знижує зв'язаність між логічними підмодулями:

```
@Module({
  imports: [
    MongooseModule.forFeature(/* ... */),
    UsersModule,
  ],
  controllers: [MatchesController],
  providers: [MatchesService, MatchesGateway],
  exports: [MatchesService]
})
export class MatchesModule {}
```

Для обробки запитів цей фреймворк утилізує фреймворк нижчого рівня Express. Він дозволяє відкривати локальний сервер та додає зручну логіку роутингу для запитів, котру NestJS імплементує в своїх контроллерах.

Для передачі інформації про статистичні данні та авторизацію було вирішено використовувати HTTP протокол. Опис роутів та функцій для обробки цих запитів знаходиться в класах підмодулів під назвою Controller:

```
@Controller('users')
export class UsersController {
    constructor(
        private usersService: UsersService
    ) {}

    @UseGuards(AuthGuard)
    @HttpCode(HttpStatus.OK)
    @Get('me')
    getMe(@Request() req) {
        return this.usersService.findOne(req.user.email);
    }

    // ...
}
```

В свою чергу передача данних про дії під час матчу, тобто синхронізація між гравцями, була реалізована за допомогою протоколу Websockets (TCP) що дозволяє підписуватись на створені топіки та отримувати повідомлення від інших вузлів. Імплементація вебсокетів була реалізована за допомогою бібліотеки socket.io, яка має інтеграцію з NestJS у вигляді нативного модулю nestjs-websockets. Опис кімнат та обробка повідомлень знаходяться у класах підмодулів під назвою Gateway:

```
@WebSocketGateway({
     // ...
})
export class MatchesGateway implements OnGatewayInit<Server> {
    @SubscribeMessage('join-room')
    joinRoom(@MessageBody() data, @ConnectedSocket() socket: Socket) {
     // ...
    }

    // ...
}
```

Основна логіка роботи відбувається в сервісах, які в свою чергу за допомогою репозиторіїв та бібліотеки MongooseORM формують запити до бази даних.

Для ініціалізації звязку з базою даних та формуванню репозиторіїв Mongoose потребує описаних схем колекцій. Це виконується в спеціальних файлах Schema:

```
@Schema()
export class User {
  @Prop()
  nickname: string;
  @Prop()
  email: string;

  @Prop()
  password: string;
```

```
  @Prop()
  rating: number;

}
export const UserSchema = SchemaFactory.createForClass(User);
```

В даних файлах описуються модельні представлення сутностей бази даних. Крім того серверна частина реалізує генерацію сторінок html розмітки з інтерфейсом користувача. Це відбувається в контроллерах за допомогою декоратору з відповідної нативного пакету фреймворку:

```
@Get('play')
@Redirect('/', 302)
async play(@Req() req) {
      return {
            url: ${req.protocol + '://' + req.get('host')}/room/${uuidv4()}
      }
}
```

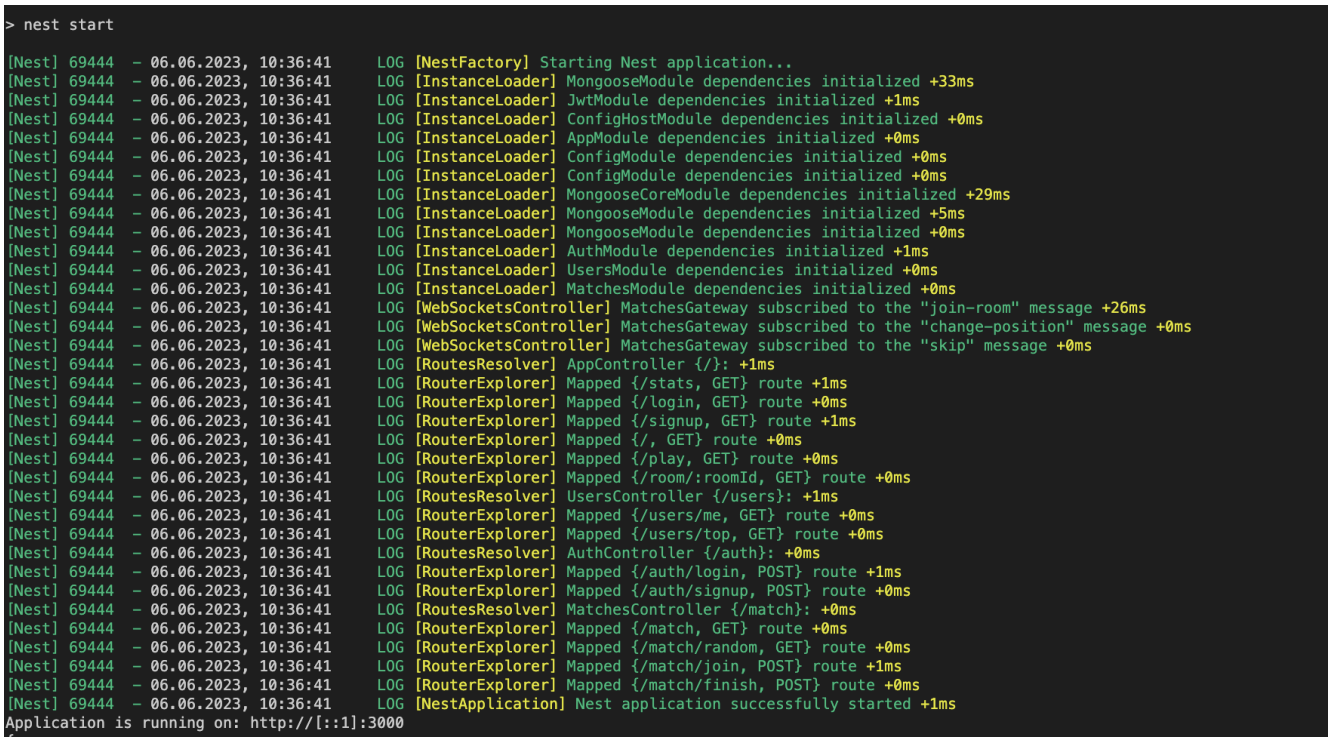


Рисунок 3.1.1 – Повідомлення про ініціалізацію усіх модулів при запуску

3.2. Реалізація бази даних

Для імплементації бази даних було обрано MongoDB. Для того щоб створити нашу базу даних ми можемо використати засіб Docker. Він полегшить процес ініціалізації бази даних, її деплой та дозволить абстрагуватись від середовища.
Ми можемо отримати образ MongoDB з реєстру контейнерів Docker Hub. Доступно багато зображень, кожне з яких представляє різні версії MongoDB або базової ОС.

Найпростішим рішенням для завантаження нашого зображення є виконання наступної команди:

```
docker pull mongo
```

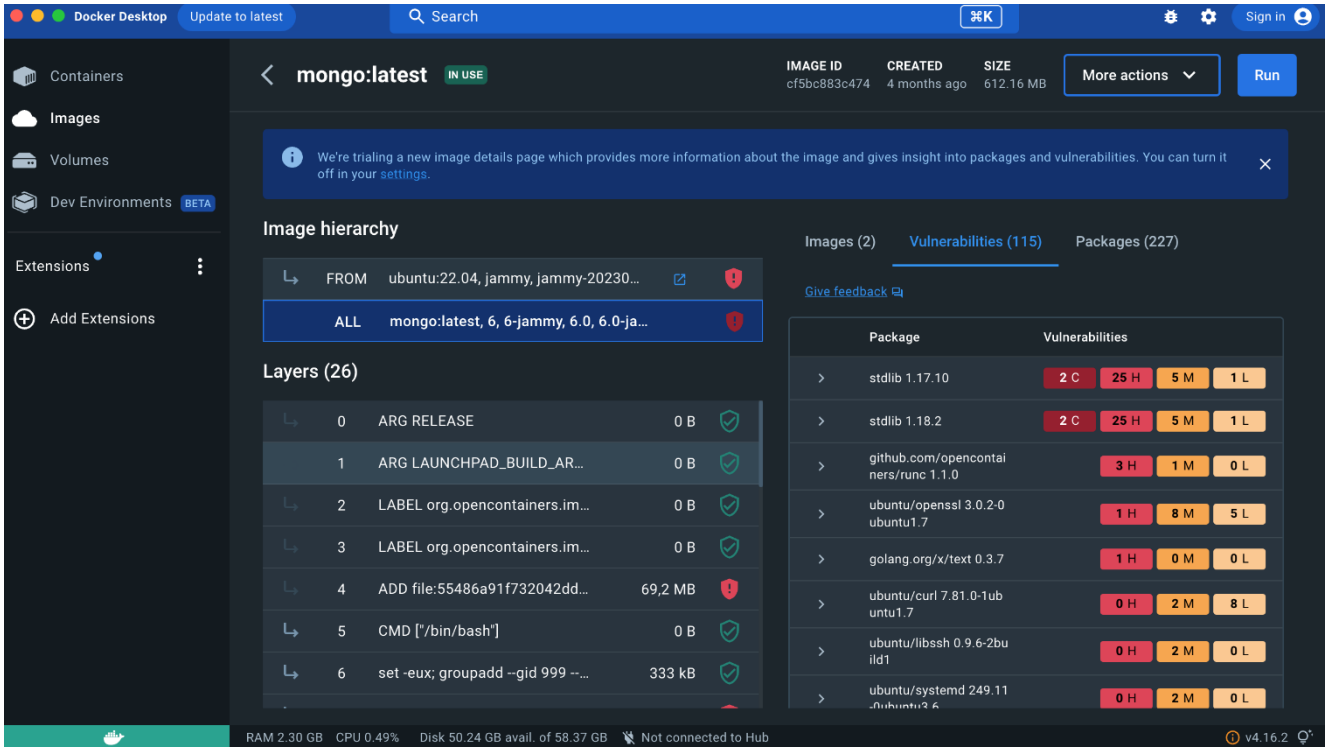


Рисунок 3.2.1 – Вигляд знімку бази даних в інтерфейсі перегляду Docker

Наведена вище команда отримає останній тег, який може бути будь-яким. Це чудово, якщо ви хочете бути передовими, але коли ви робите реалістичне розгортання, ви, ймовірно, хочете контролювати інформацію про версії. Замість цього ви можете вказати тег так:

```
docker pull mongo:4.0.4
```

Після цього нам потрібно створити контейнер відповідний цьому знімку. У своїй найпростішій формі та тому, що описано в документації Docker Hub, ми можемо виконати таку команду:

```
docker run --name mongodb mongo:4.0.4
```

Ця команда працює, але потенційно є кілька проблем. Що ми робимо, так це запускаємо контейнер і називаємо його mongodb. Для того підключитися до нього, нам доведеться зробити це з іншого примірника контейнера. Іншими словами, ми не зможемо підключитися до нього з нашого хост-комп'ютера чи сервера. Замість цього потрібно виконати наступне:

```
docker run -d -p 27017-27019:27017-27019 --name mongodb mongo:4.0.4
```

Команда вище запустить контейнер у відокремленому режимі або у фоновому режимі. Ми також зіставляємо порти контейнера з портами хоста, щоб мати доступ до бази даних із програми на рівні хоста, якщо захочемо.

Після цього потрібно запустити наш серверний модуль щоб він за допомогою сервісів Mongoose проаналізував наші схеми сутностей та створив або оновив відповідну базу даних та колекції в ній.

Адмініструвати нашу базу даних ми можемо використовуючи графічний інтерфейс безплатної програми MongoDB Atlas або використовуючи мову запитів API бази даних.

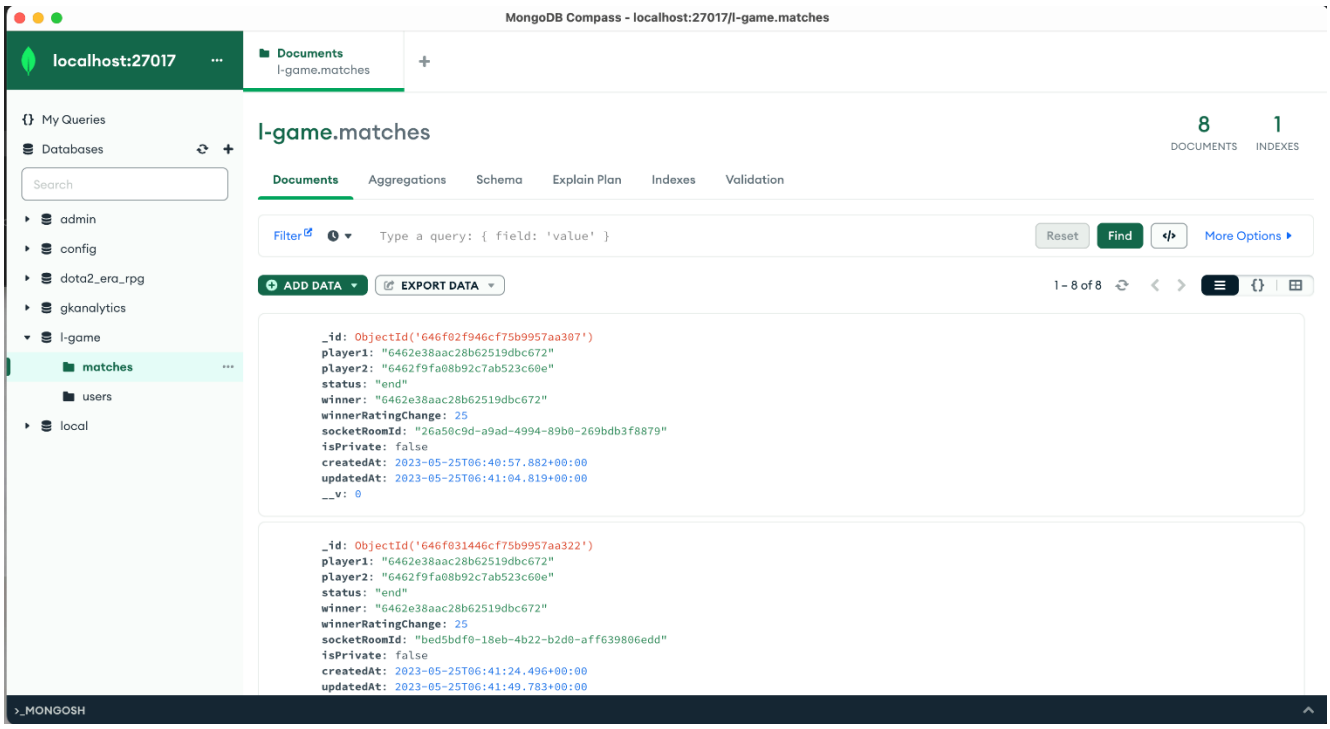


Рисунок 3.2.2 – Вигляд бази даних в інтерфейсі MongoDB Atlas

3.3. Інструкції використання

Після реалізації головного функціоналу було сформовано детальну інструкцію щодо використання веб сервісу. Інструкція з використання:

Як увійти в обліковий запис:

а) якщо ви не авторизовані виконайте такі дії:

- знайдіть кнопку входу на сайті;
- натисніть кнопку входу, щоб отримати доступ до сторінки входу;
- введіть свої облікові дані для входу (ім’я користувача або адресу електронної пошти та пароль) у відповідних полях;
- натисніть кнопку входу, щоб авторизуватися та отримати доступ до свого облікового запису.

б) якщо у вас немає облікового запису, виконайте такі дії:

- знайдіть на сайті кнопку реєстрації;
- натисніть кнопку реєстрації, щоб перейти на сторінку реєстрації;
- заповніть необхідні поля, вказавши бажане ім’я користувача, адресу електронної пошти та пароль;

– натисніть кнопку реєстрації, щоб створити обліковий запис;

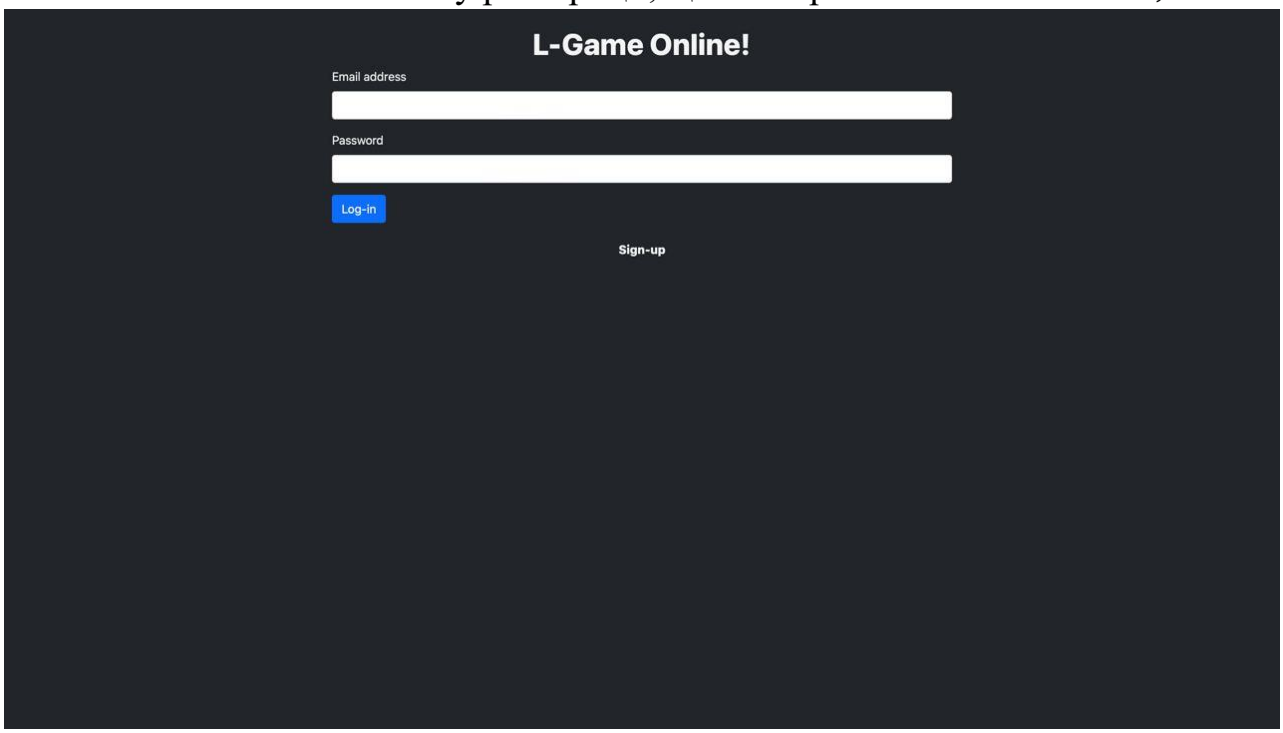


Рисунок 3.3.1 – Екран входу користувача

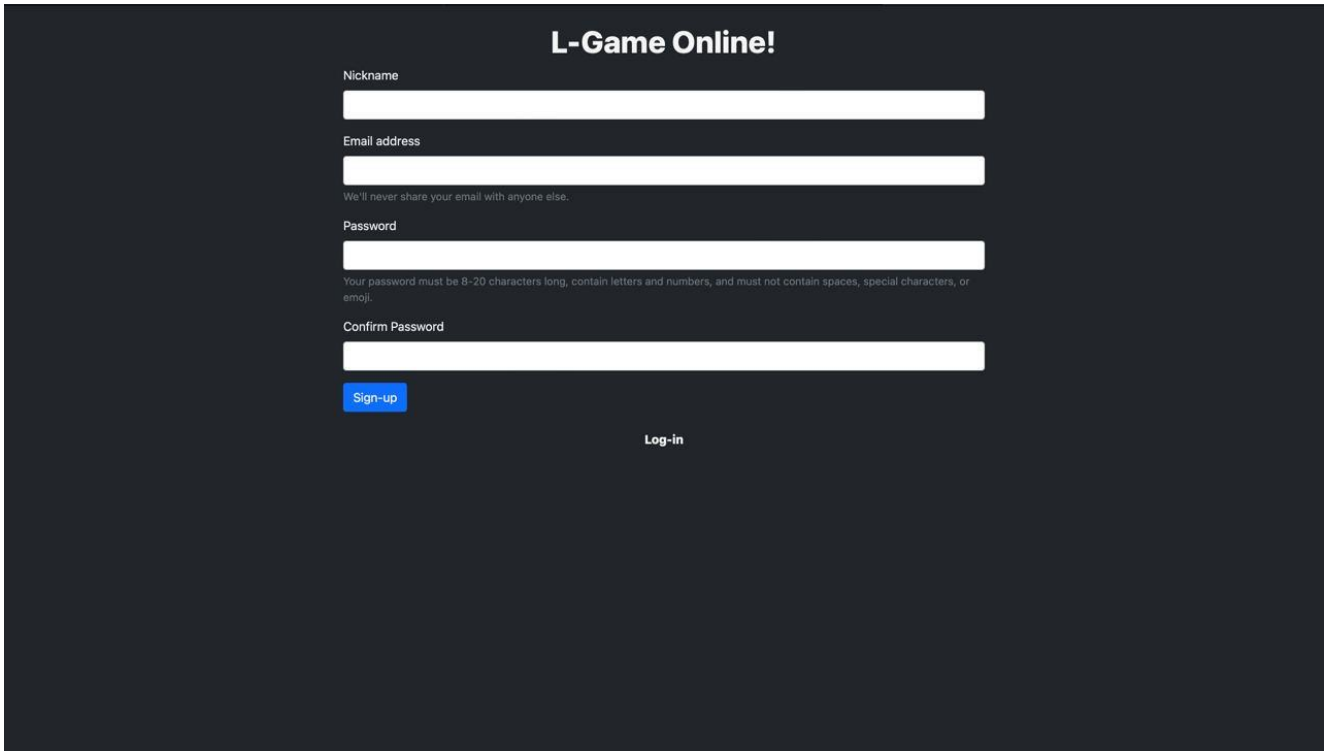


Рисунок 3.3.2 – Екран реєстрації користувача

Як почати грати:

а) щоб приєднатися до наявної кімнати, виконайте такі дії:

– знайдіть кнопку гри на веб-сайті;

– натисніть кнопку гри, щоб підключитися до вже створеної кімнати;
– зачекайте, доки гра завантажиться, і вас з'єднають з гравцями в кімнаті.

б) щоб створити нову кімнату та запросити інших, виконайте такі дії:
– знайдіть кнопку створити кімнату на веб-сайті;
– натисніть кнопку створити кімнату, щоб розпочати створення кімнати;
– налаштуйте параметри кімнати, наприклад режим гри або назву кімнати, якщо це можливо;
– після встановлення потрібних параметрів натисніть кнопку створити, щоб створити кімнату;
– поділіться посиланням на кімнату з друзями або іншими гравцями, щоб запросити їх приєднатися до кімнати;
– коли гравці приєднаються, гра почнеться автоматично;

в) якщо ви отримали спільне посилання на кімнату, виконайте такі дії:
– скопіюйте надане вам спільне посилання;
– вставте посилання в адресний рядок браузера;
– натисніть Enter, щоб підключитися та приєднатися до гри.

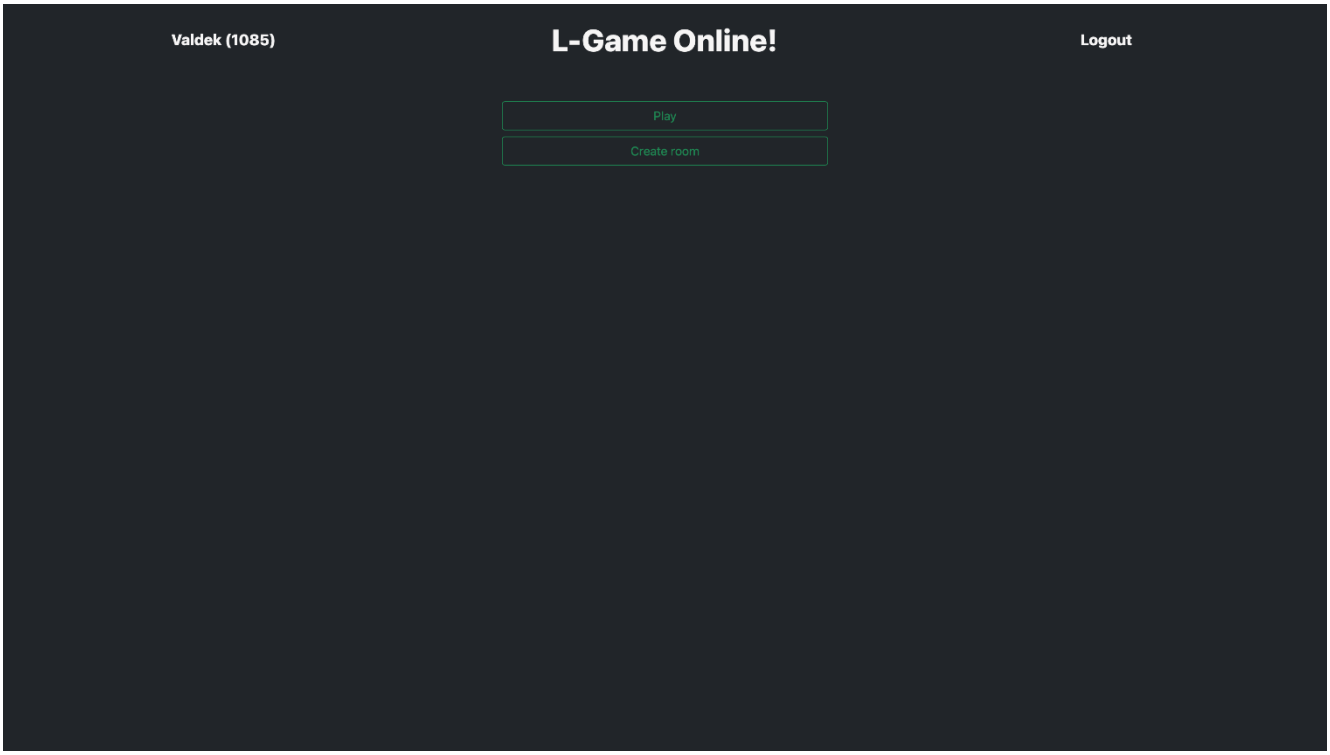


Рисунок 3.3.3 – Екран меню

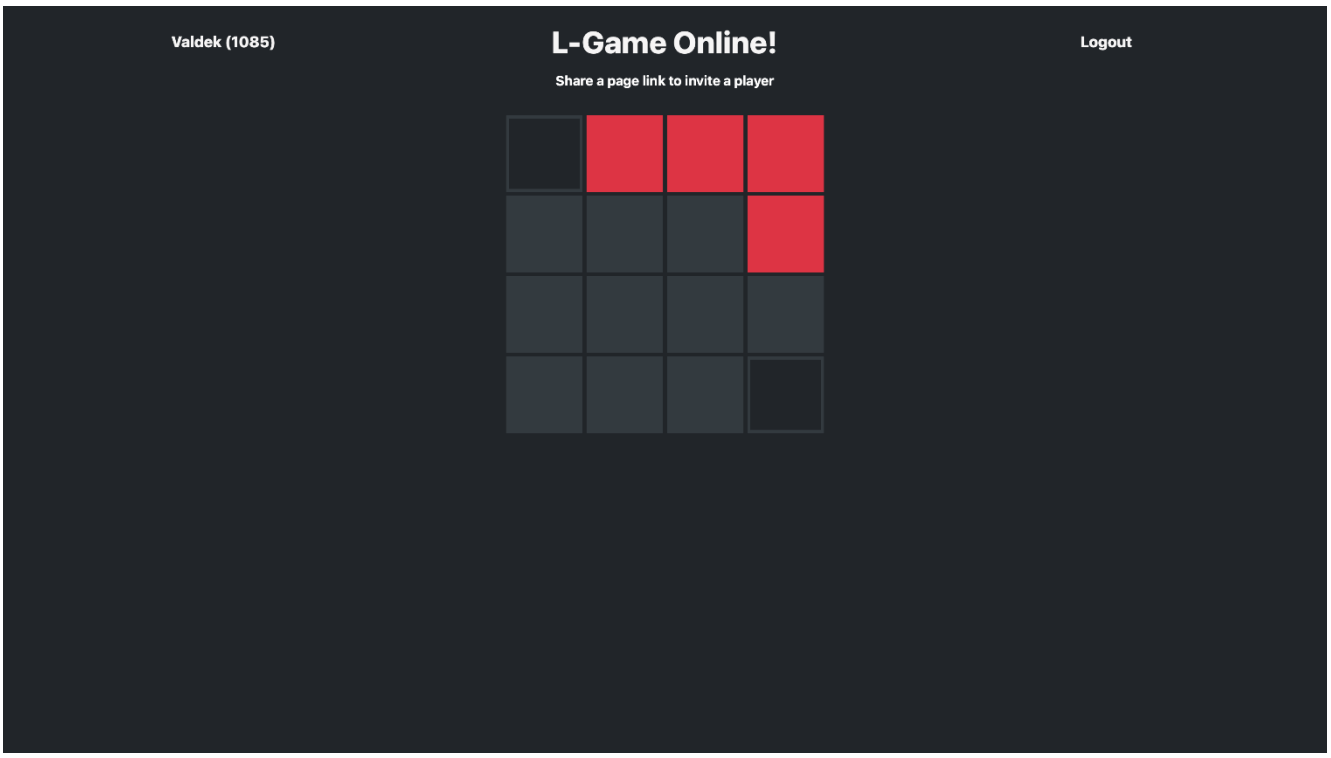


Рисунок 3.3.4 – Екран гри в режимі очікування на опонента

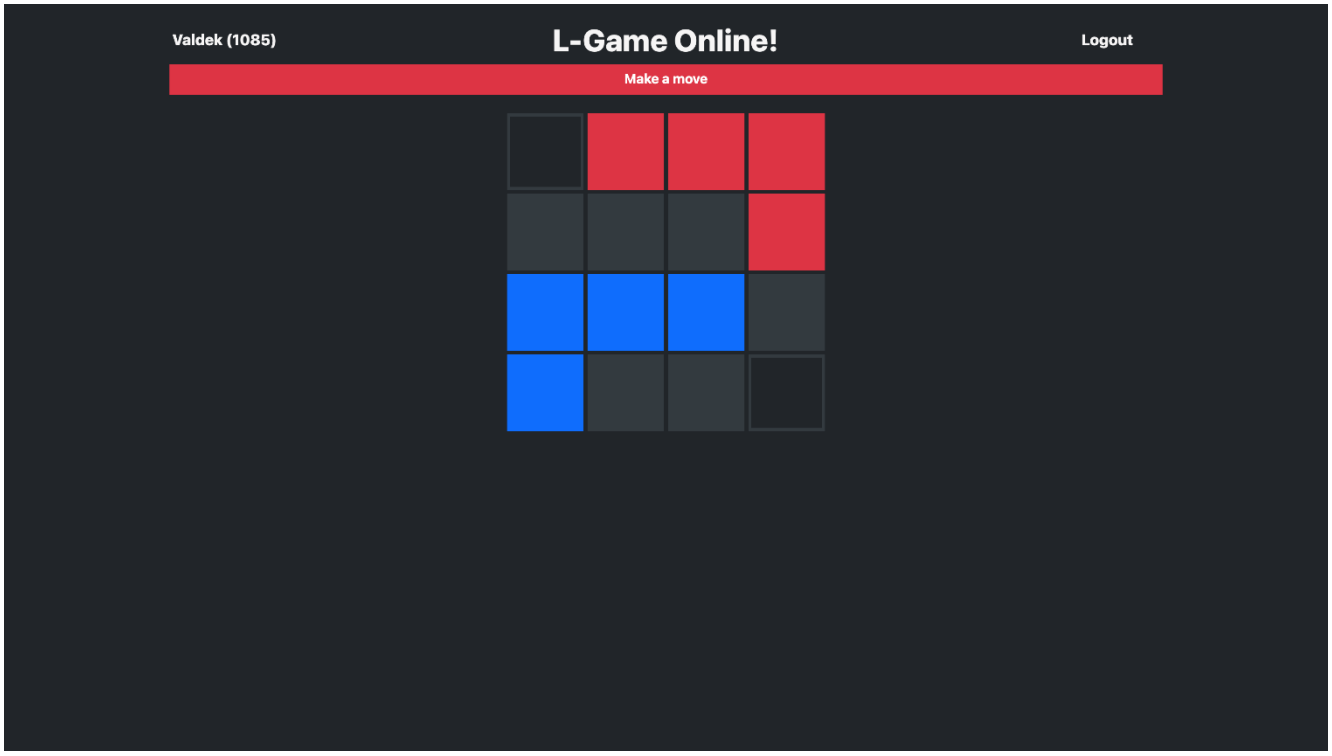


Рисунок 3.3.5 – Екран гри під час матчу

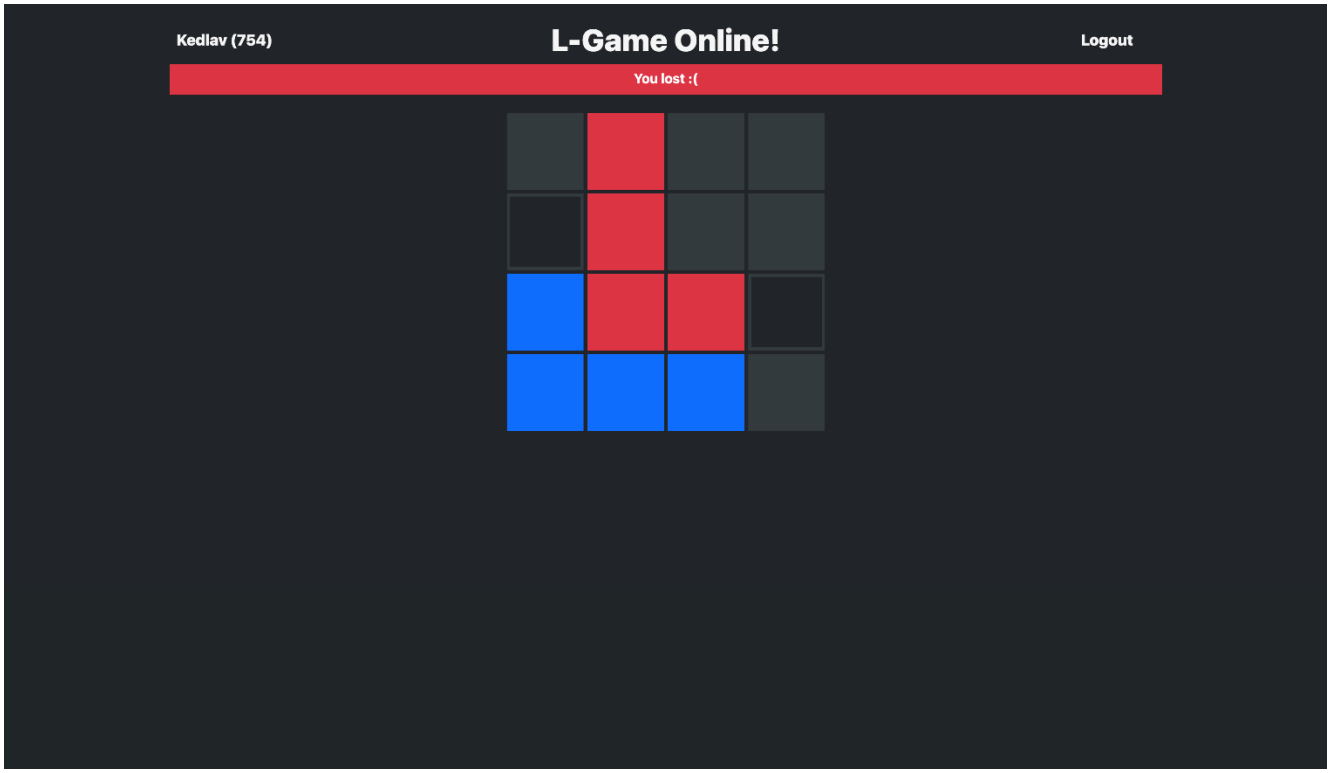


Рисунок 3.3.6 – Екран гри після завершення матчу

Правила гри:

Гра відбувається на полі 5 на 5 клітинок. В кожного гравця своя L-подібна фігура що розміщується на полі. Також на полі розміщується 2 жетони розміром з клітинку, рухати їх можуть обидва гравці. Коли починається хід гравця, він ПОВИНЕН пересунути свою Л-фігурку в інше місце, дозволяється довільно повертати фігуру. Після цього гравець може пересунути також один жетон, або пропустити цей етап і закінчити свій хід. Гравці ходять по черзі. Гра закінчується тоді коли один з гравців немає доступних варіантів для переміщення своєї фігури - такий гравець програє. Насправді правила гри можуть відрізнятися в залежності від країни у якій проходить гра, проте наведений список правил є одним з найбільш вживаних та загальним для більшості країн.

Також користувач може переглянути статистику зіграних ігор.

Щоб переглянути свою статистику, виконайте такі дії:

– знайдіть назву свого облікового запису у верхній частині заголовка сайту;
– натисніть на назву свого акаунту, щоб отримати доступ до статистики.

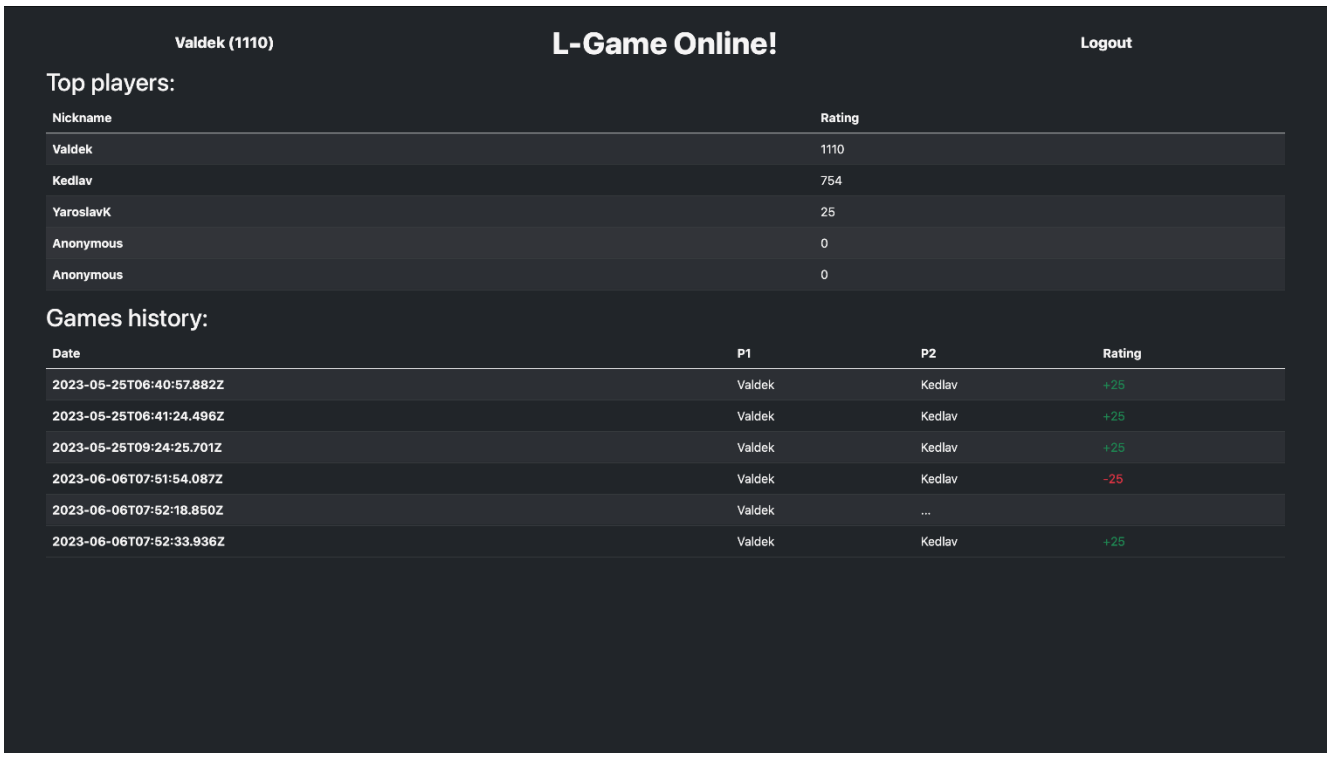


Рисунок 3.3.7 – Екран перегляду статистики

Ці інструкції містять вичерпний посібник для користувачів з навігації та використання веб сервісу. Використовуючи дану ісntрукцію користувач зможе зрозуміти як користуватись програмою та виконувати ті чи інші, дії, наприклад: як виконувати авторизацію, як грати в гру, як підключитись до кімнати для гри з другом та інше.

3.4. Технічні характеристики

Було сформовано вимоги до комплектації комп'ютера на якому рекомендовано запускати клієнтський код:

– наявність миші (дотик сенсорного екрану не зчитується в процесі гри);

– екран більший 300x300px, для того щоб компоненти могли масштабуватись правильно;

– підключення до мережі інтернет;

– підтримка браузерів;

– процесор що зможе обслуговувати середню за навантаженням вкладку

браузера та постійні операції з мережею;

– інтегрована або дискретна відеокарта;

– приблизно 400мб вільної оперативної пам'яті (в залежності від браузера).

Крім того важливо зазначити що деякі використані бібліотеки та функції не підтримуються на старих браузерах, тому було складено таку таблицю сумісності з браузерами та платформами:

	Chrome	Firefox	Opera	Safari	Edge
Android	v14+	v22+	v5.2+	-	v10+
iOS	v12+	v22+	-	v9+	-
Windows 10 Mobile	-	-	-	Не підтр.	v10+
Mac	v45+	v38+	v30+	v9+	-
Windows	v46+	v36+	v30+	Не підтр.	v12+

3.5. Тестування

Головним методом тестування веб-сервісу L-game було обрано тестування методом «чорної скриньки». Цей метод тестування працює з вже працюючою програмою, з його інтерфейсом та зовнішніми сервісами. Таке тестування дозволяє протестувати як і сам застосунок так і правильність його функціонування разом з іншими сервісами, одночасно тестуючи всі його складові.

Щоб цей метод був ефективним було створено список з тест-кейсів – найважливіших та найймовірніших сценаріїв подій роботи з веб-сервісом.

Тест-кейси було сформовано в форматі відображеному в Таблиці А.2:

Після формування тест-кейсів було визначена стратегія тестування. Кожна функція, тобто реалізація поставленої вимоги повинна розроблятися наскрізно і повністю. Після закінчення імплементації функції починається її тестування. Тестувальник проводить всі тест-кейси і порівнює результати проведеної роботи з

очікуваними результатами тест-кейсу. Якщо вони не збігаються тестувальник повинен створити картку з посиланням на тест-кейс, де опише результат який отримав він та вкаже на його відмінності. Крім того в баг-репорті бажано вказувати допоміжні інформацію про стан продукту під час тестування щоб полегшити виправлення помилок.

Після повного проходження по всім тест-кейсам функції та створення баг-репортів проводиться виправлення невідповідностей в веб-сервісі проводиться ще одна ітерація тестування і виправлення, допоки всі невідповідності не будуть виявлені та виправленні.

3.6. Результат програмної реалізації та тестування

Отже, в результаті виконання програмної реалізації створено повністю функціональний і надійний веб-сервіс. Було створено тестувальні практики для того щоб впевнетись що продукт відповідає всім вимогам та працює правильно.

Детальний дизайн модулів програми, включаючи клієнтський модуль, серверний модуль і модуль бази даних, було перетворено на реальний код. Процес впровадження передбачав використання відповідних технологій і фреймворків, таких як HTML, CSS, JavaScript, Python і Flask, щоб забезпечити ефективну розробку та сумісність на різних платформах.

Інтерфейс користувача розроблено з великою увагою до деталей, зосереджуючись на створенні інтуїтивно зрозумілого та візуально привабливого дизайну. Користувачі можуть легко переміщатися веб-сайтом, отримувати доступ до таких функцій, як вхід до облікового запису, грати з незнайомцями чи друзями та легко переглядати свою статистику.

На етапі впровадження було проведено ретельне тестування, щоб переконатися в надійності та функціональності веб-сервісу. Для виявлення та вирішення будь-яких потенційних проблем або помилок використовувалися різні

методології тестування, такі як модульне тестування, інтеграційне тестування та тестування прийнятності користувача.

Успішно реалізована функція мультиплеєра, яка дозволяє користувачам грати з незнайомцями або запрошувати друзів. Спілкування в режимі реального часу між гравцями забезпечується через Websockets, що забезпечує синхронізацію ігрового процесу та захоплюючий інтерактивний досвід.

Функцію відстеження статистики було інтегровано у веб-сервіс, що дозволяє користувачам контролювати продуктивність свого ігрового процесу. Користувачі можуть переглядати різноманітну статистику, таку як кількість зіграних ігор, перемоги, поразки та інші показники, підвищуючи змагальний аспект гри.

Підсумовуючи, етап впровадження та тестування програмного забезпечення пройшов успішно. Веб-сервіс є повністю функціональним, включаючи всі розроблені функції та функції. Для забезпечення надійності та задоволеності користувачів було проведено ретельне тестування.

ВИСНОВКИ

Під час вивчення предметної області було проведено масштабне дослідження, щоб отримати глибоке розуміння гри L-пазл та її привабливості для користувачів. Це дослідження допомогло визначити ключові функції та функції, які слід включити до веб-сайту. Аналізуючи існуючі платформи та вивчаючи вподобання користувачів, команда проекту встановила контрольні показники та отримала цінну інформацію про очікування користувачів, переконавшись, що веб-сайт відповідає галузевим стандартам і задовольняє потреби цільової аудиторії.

Фаза проектування проекту була надзвичайно важливою, оскільки вона передбачала перетворення визначених вимог і специфікацій на інтуїтивно зрозумілий і зручний інтерфейс. Було вжито ретельного підходу, щоб гарантувати, що дизайн веб-сайту був візуально привабливим, простим у навігації та чуйним для різних пристроїв.

Вибрану архітектуру та технології було ретельно оцінено, щоб переконатися, що вони забезпечать ефективну, безпечну та масштабовану платформу для користувачів, щоб насолоджуватися грою L-пазл. Дизайнерські рішення приймалися в тісній співпраці із зацікавленими сторонами проекту, враховуючи їхні відгуки та узгоджуючи їх із цілями проекту.

Наступний етап впровадження включав перетворення концепцій дизайну на повністю функціональний веб-сайт. Команда розробників використала свій технічний досвід, щоб втілити задуманий веб-сайт у життя. Вони успішно інтегрували такі функції, як реєстрація користувачів, функція входу, ігрова механіка та статистичне відстеження. Завдяки ретельному кодуванню та ретельному тестуванню було реалізовано основні функції веб-сайту, що дозволило користувачам входити в систему, грати з незнайомцями, запрошувати друзів і переглядати статистику своїх ігор. Процес впровадження дотримувався найкращих галузевих практик, забезпечуючи чистий код, ефективні алгоритми та бездоганну інтеграцію різних компонентів.

Етап тестування відіграв ключову роль у перевірці функціональності та ефективності веб-сайту. Для оцінки стабільності, швидкості реагування та загального досвіду роботи веб-сайту було застосовано ретельні процедури тестування. Результати тестування дали цінну інформацію та підтвердили, що веб-сайт працював надзвичайно добре, відповідаючи вимогам і очікуванням, викладеним у Технічній специфікації. Основні функції, такі як переміщення L-подібних фігур, взаємодія з пішаками та покрокова гра, були ретельно перевірені та функціонують належним чином.

Незважаючи на загальний успіх проекту, етап тестування також виявив кілька некритичних проблем, які можна було б удосконалити. Ці проблеми зосереджені насамперед навколо незначних невідповідностей в інтерфейсі користувача, випадкових затримок у спілкуванні в реальному часі між гравцями та конфліктів сумісності з певними веб-браузерами. Хоча ці проблеми не вплинули суттєво на загальну функціональність і зручність використання веб-сайту, їх усунення, безсумнівно, покращить взаємодію з користувачем і забезпечить більш досконалий кінцевий продукт.

Підсумовуючи, розробка «Веб-сервісу для гри в L-головоломку» мала успіх. Комплексне вивчення предметної області, ретельна робота над дизайном, ефективне впровадження та ретельне тестування сприяли створенню привабливого та функціонального веб-сайту. Проект успішно досяг своїх цілей, надавши користувачам платформу для насолоди грою L-пазл, взаємодії з іншими гравцями та відстеження їх ігрової статистики. Виявлені некритичні проблеми будуть вирішені в майбутніх оновленнях для подальшого покращення взаємодії з користувачем і забезпечення вдосконаленої та безперебійної веб-служби. «Веб-сайт для гри в L-головоломку» має величезні перспективи та є свідченням відданості, майстерності та досвіду команди розробників.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Л. П. Бедратюк. Дипломний проект: методичні вказівки щодо його виконання для студентів спеціальності 121 «Інженерія програмного забезпечення» / Л. П. Бедратюк, Г. І. Радельчук, Ю. В. Форкун. О. М. Яшина – Хмельницький: ХНУ, 2020. 77с.
2. Modern web development. URL: https://www.udemy.com/course/mordern-web-development-course-beginner-to-advanced . (дата звернення – 10.04.2023).
3. HTML games development. URL: https://www.w3schools.com/graphics/game_intro.asp. (дата звернення – 02.03.2023).
4. HTML chessboard tutorial. URL: https://www.chessjournal.com/how-to-make-chess-board-css. (дата звернення – 11.04.2023).
5. NodeJS server. URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction. (дата звернення – 27.04.2023).
6. NestJS quick start. URL: https://dev.to/kalashin1/getting-started-with-nestjs-1p1d. (дата звернення – 06.05.2023).
7. SSR guide. URL: https://www.ntegral.com/insights/server-side-rendering-ssr-angular-universal-nest-js. (дата звернення – 24.04.2023).
8. Bootstrap quick start. URL: https://www.w3schools.com/bootstrap/bootstrap_get_started.asp. (дата звернення – 10.05.2023).
9. Jason Gregory. Game Engine Architecture. A K Peters/CRC Press. 3rd edition 2018. 1240p . (дата звернення – 06.04.2023).
10. Gerald Farin, Dianne Hansford. Practical Linear Algebra. Chapman and Hall/CRC. 4th edition 2021. 590c.
11. Bootstrap quick start. URL: https://oscarotero.com/jquery/. (дата звернення – 20.05.2023).
12. NPM tutorial. URL: https://nodesource.com/blog/an-absolute-beginners-guide-to-using-npm/. (дата звернення – 13.04.2023).

13. MongoDB documentation. URL: https://www.mongodb.com/docs/ . (дата звернення – 21.03.2023).

14. Mongoose quick start. URL: https://mongoosejs.com/docs/ . (дата звернення – 03.04.2023).

15. Websockets. URL: https://developer.mozilla.org/ru/docs/Web/API/WebSockets_API . (дата звернення – 10.04.2023).

16. Socket.io documentation. URL: https://socket.io/docs/v4/ . (дата звернення – 05.03.2023).

17. NestJS injections guide. URL: https://docs.nestjs.com/providers . (дата звернення – 21.03.2023).

18. L-puzzle Game Official Documentation. URL: http://boardgame.bg/project%20l%20%20rules.pdf. (дата звернення - 01.04.2023)

19. "The Impact of Online Gaming on User Engagement" Research Paper. URL: https://www.sciencedirect.com/science/article/pii/S0148296321004409. (дата звернення - 02.04.2023)

20. Journal of Game Studies. URL: https://www.gamestudies.org/0101/editorial.html. (дата звернення - 03.04.2023)

21. GameDev Insider. URL: https://gamedevinsider.com/breakdown-of-different-programming-languages-used-in-game-development/. (дата звернення - 05.04.2023)

22. Game Design Academy (Game Design Tutorials). URL: https://gamesacademy.com.ua/. (дата звернення - 08.04.2023)

23. Game Developers Conference. URL: https://gdconf.com/. (дата звернення - 09.04.2023)

24. Game Informer. URL: https://www.gameinformer.com/. (дата звернення - 13.04.2023)

25. Game Analytics. URL: https://gameanalytics.com/. (дата звернення - 15.04.2023)

26. Gamasutra. URL: https://gameworldobserver.com/2021/08/23/gamasutra-getting-renamed-to-game-developer-and-moving-to-new-site. (дата звернення - 19.04.2023)

27. Game Accessibility Guidelines. URL: https://gameaccessibilityguidelines.com/. (дата звернення - 22.04.2023)

28. UX Design for Gaming Websites. URL: https://dribbble.com/tags/gaming%20website. (дата звернення - 24.04.2023)

29. GamePro. URLhttps://gamepro.zone/. (дата звернення - 25.04.2023)

30. Mario Casciaro. Node.js Design Patterns. Packt Publishing. 2nd edition 2016. 777с.

31. Jonathan Wexler. Programming with Node.js Manning. First Edition 2019. 480 с.

32. Andrew Mead. Advanced Node.js Packt Publishing. 2018. 592с.

33. Thomas Hunter II. Distributed Systems with Node.js O'Reilly Media. 1st edition 2020. 377с.

34. Mr Shaun Michael Stone. Automating with Node.js Independently published 2018. 196с.

35. Azat Mardan. Practical Node.js Apress. 2nd ed. edition 2018. 529с.

36. Valentin Bojinov. RESTful Web API Design with Node.js Packt Publishing. 2nd edition 2016. 236с.

37. Bethany Griggs. Node Cookbook Packt Publishing. 4th ed. edition 2020. 512с.

38. Ray Yao. NODE.JS in 8 Hours Independently published 2020. 128с.

39. David Herron. Node.js Web Development Packt Publishing 2020. 760с.

40. Game Development World Championship URL: https://thegdwc.com/. (дата звернення - 27.04.2023)

ДОДАТОК А
(обов’язковий)

ТАБЛИЦІ

Назва	Розробник	Переваги	Недоліки
Greenfoot.org	Greenfoot Education Community	Орієнтація на освітні цілі, широкий вибір ресурсів	Немає режиму для гри проти комп'ютера/гравців, обмежена функціональність, застарілий дизайн
L-game from Vemodalen	Vemodalen	Сучасний дизайн, можливість грати проти AI/гравців	Обмежена підтримка Android, відсутність статистики
hwwmath.looiwenli.com	Looi Wenli	Режими PvP і PvE, кросплатформна сумісність	Застарілий дизайн, відсутність статистики

Таблиця А.1 – Порівняння вже існуючих програмних рішень

Номер	Пріоритет	Модуль	Підмодуль	Вхідні дані	Очікуваний результат
3	A	Auth	Registration	Очистити всі cookies	
				Створити або мати зареєстрований акаунт в базі даних	
				Перейти в головне меню сервісу	Відображається сторінка головного меню
				Нажати на кнопку Login	Відображається сторінка входу в акаунт
				Ввести в поле Email електронну почту акаунту для входу	
				Ввести в поле Password пароль акаунту для входу	В полях відображаються ввведені дані, значення поля паролю приховано зірочками
				Нажати на кнопку Login	Перенесено на сторінку головного меню, не відображається ніяких повідомлень про помилку. У лівому верхньому кутку відображається нікнейм акаунту в який було здійснено вхід

Таблиця А.2 – Приклад формату тест-кейсу

ДОДАТОК Б
(обов’язковий)

ДІАГРАМИ

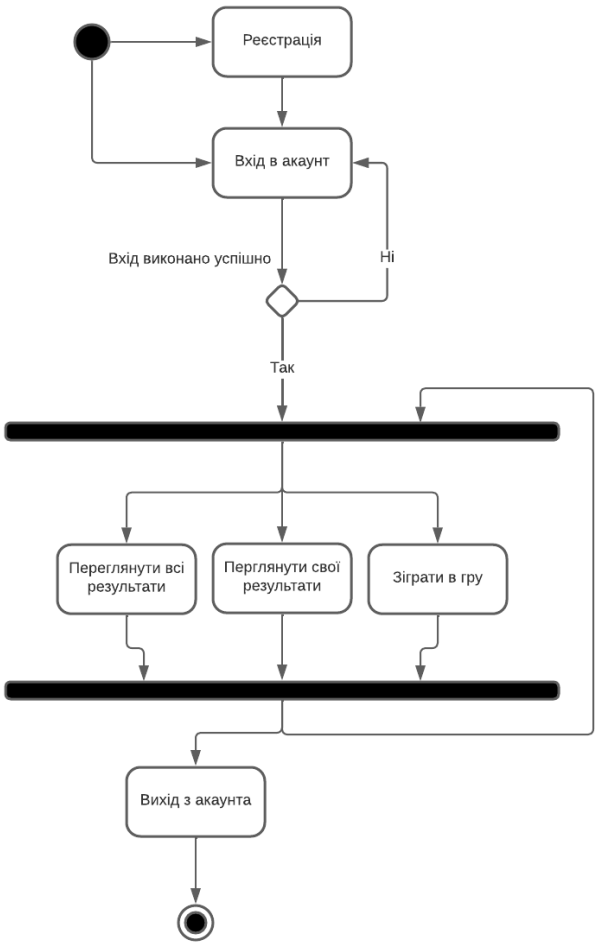


Рисунок Б.1 – Діаграма станів користувацького інтерфейсу

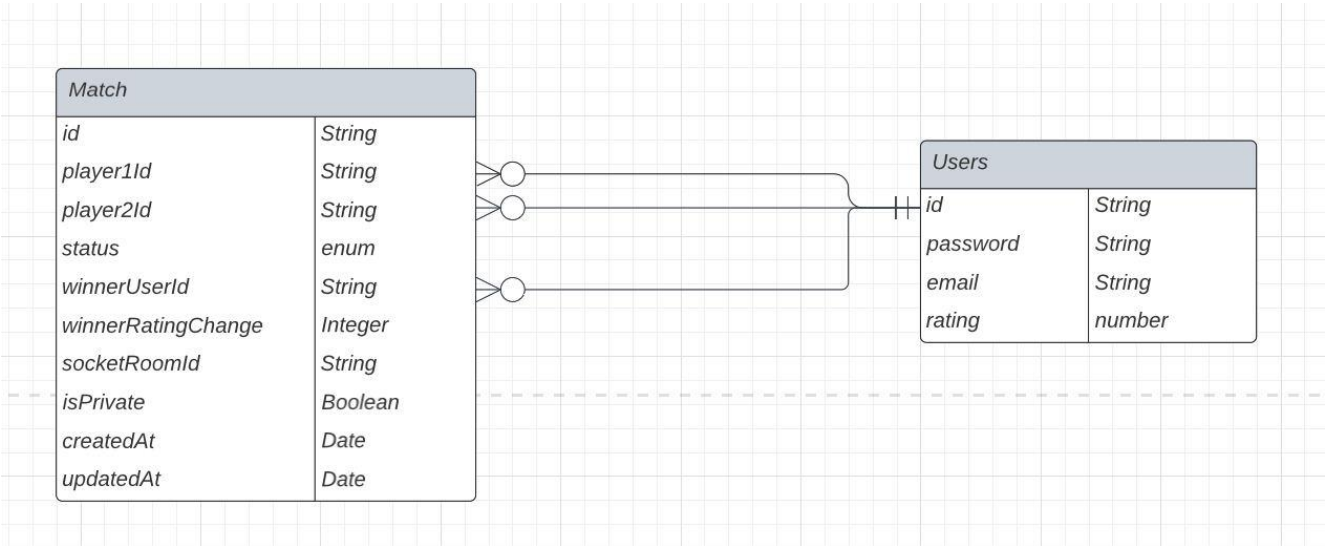


Рисунок Б.2 – Діаграма бази даних

ДОДАТОК В
(обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

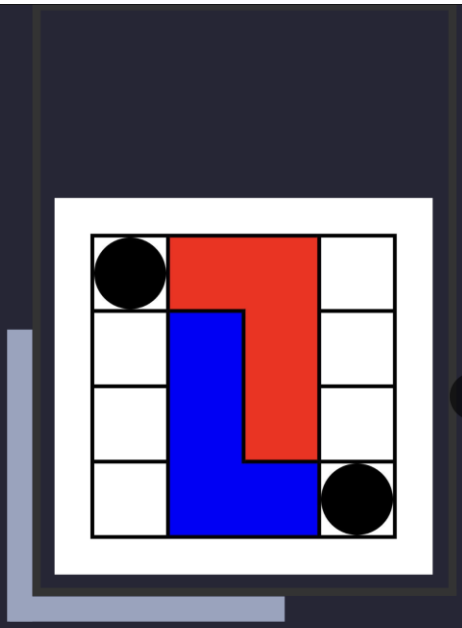
Вступ

Суть гри та актуальність теми

L-Game — це класична гра-головоломка, яку винайшов Едвард де Боно в 1960 році.

Мета гри полягає в тому, щоб перемістити свою Л-подібну фігуру найбільш вигідним для себе чином, не даючи противнику зробити те саме.

Гра розкриває стратегічні навчики гравців, уміння прораховувати дії наперед та читати противника.

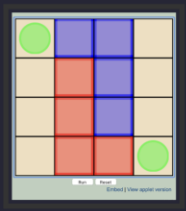


Доцільність теми

- **Низький рівень входу**
  У гри доволі прості правила, яких можна навчитись за кілька хвилин. Проте найцікавіше починається коли гравець вже зіграв кілька партій і починає мислити стратегічно щоб перемагати частіше.
- **Гра не займає багато часу**
  В залежності від рівня гравців партії можуть тривати від 10хв до нескінченності, проте раунди з випадковими опонентами рідко тривають більше 20хвилин
- **Проста структура гри**
  Простота функціоналу гри дозволяє максимально зконцентруватись на зручності у користуванні та розробити нові приємні доповнення що будуть оживляти гру
- **Мала кількість альтернатив**
  Попри простоту цієї головоломки, вона так і не стала популярною. Через це засобів для гри в інтернет дуже мало, а хороших продуктів взагалі одиниці.

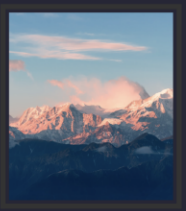
Приклади

уже існуючих реалізацій



Greenfoot.org
- немає режиму гри PvP та PvE
- обмежена функціональність
- застарілий дизайн

L-game від Vemodalen
+ Сучасний дизайн
- Підтримка тільки Android
- Відсутність статистики, рейтингу і т.п.



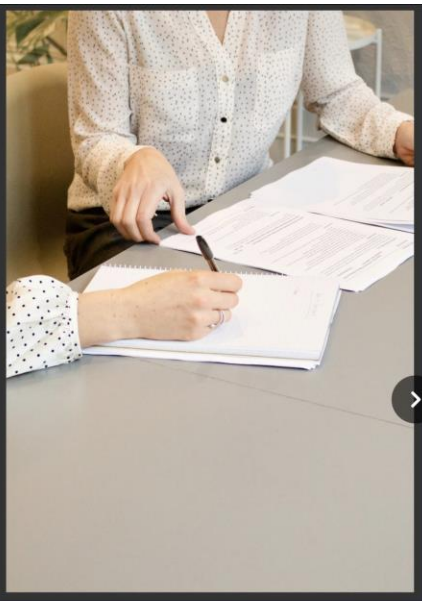
hwwmath.looiwenli.com
+ Можливість PvP та PvE гри
+ Кросплатформенність
- Застарілий дизайн
- Відсутність статистики

Постановка задачі

Потрібно:

Проаналізувавши існуючі реалізації, створити продукт, що буде реалізувати гру-головоломку L-game у найкращий спосіб. Продукт повинен

- **Бути кросплатформенним**
  Користувачі повинні мати змогу користуватись додатком як на смартфоні, так і на компютері.
- **Мати сучасний дизайн**
- **Зберігати результати гри**
  Відображати прогрес кожного користувача та виділяти найздібніших гравців
- **Дозволяти грати з друзями**
  Користувач повинен мати змогу запросити друга до гри
- **Залишатись простим**
  Потрібно зберегти головну особливість цієї головоломки - простоту

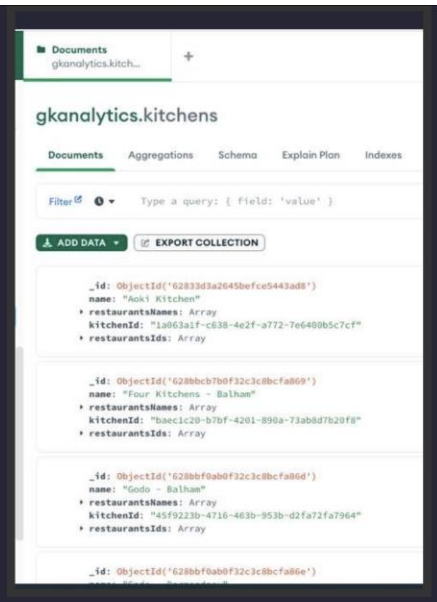


Проектування та розробка

Вибір БД

Як головну базу даних було обрано mongoDB

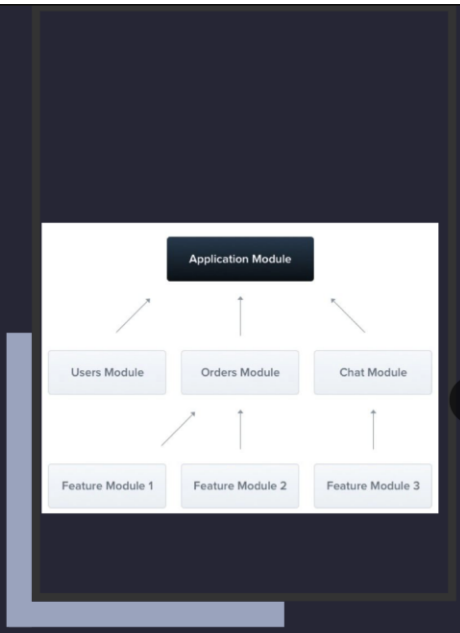
Через просту структуру інформації mongoDB буде найзручнішим та найефективнішим вибором



Вибір мови програмування

та фреймворку

Для розробки було обрано мову програмування JavaScript для клієнтської частини, та NodeJS для серверної. Така комбінація дозволяє легко обєднувати обидві частини додатку у загальну кодову базу, а за допомогою фреймворку NestJS ми можемо легко створити застосунок тримаючи всі його частини у відповідних модулях, зберігаючи порядок та зрозумілість між компонентами



Вибір транспорту для даних

Для реалізації завдання було використано два способи звязку між клієнтом та сервером:

HTTP/S протокол для одноразового та нечастого передавання інформації (запит на отримання статистики, завантаження згенерованої HTML розмітки з серверу, ауторизація та ін.)

TCP (WebSockets) протокол звязку для активного обміну повідомленнями між клієнтами та сервером під час гри у головоломку.

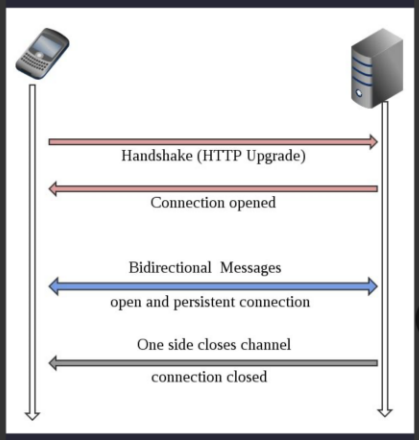


Схема структури фронтенду додатку

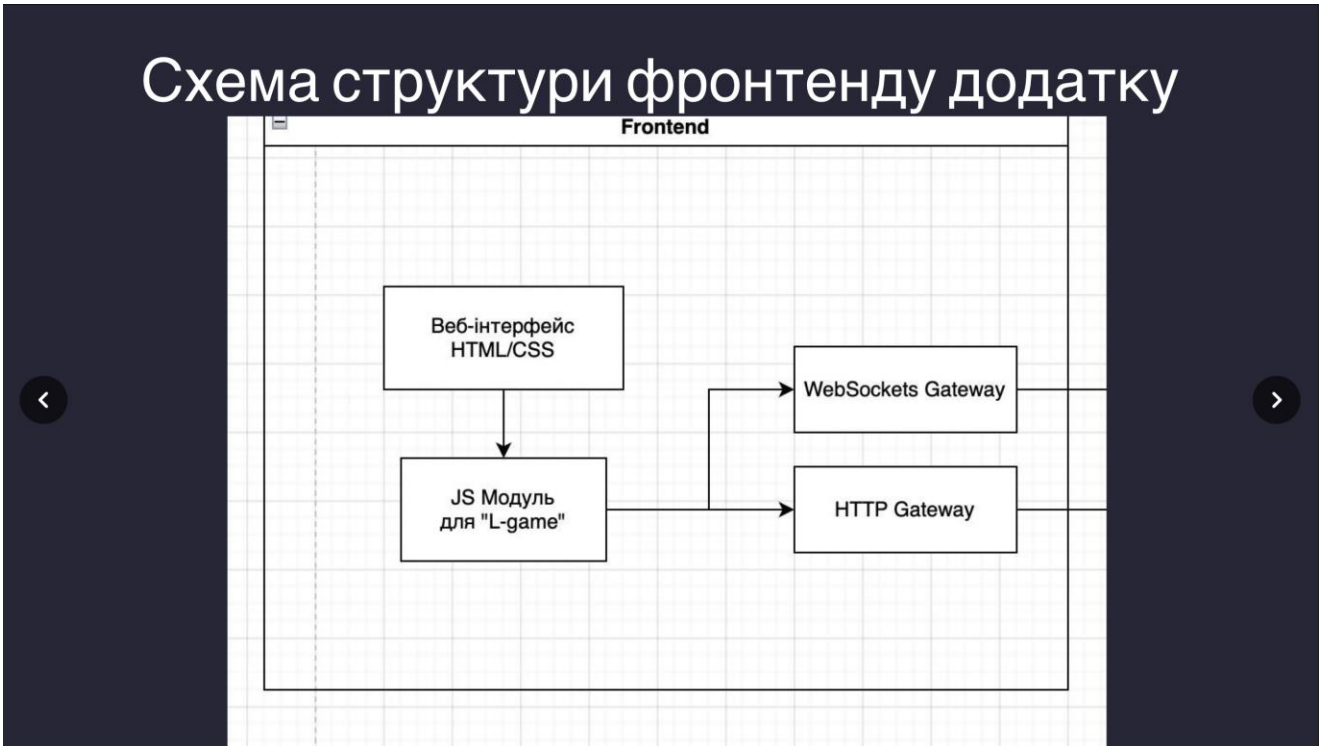
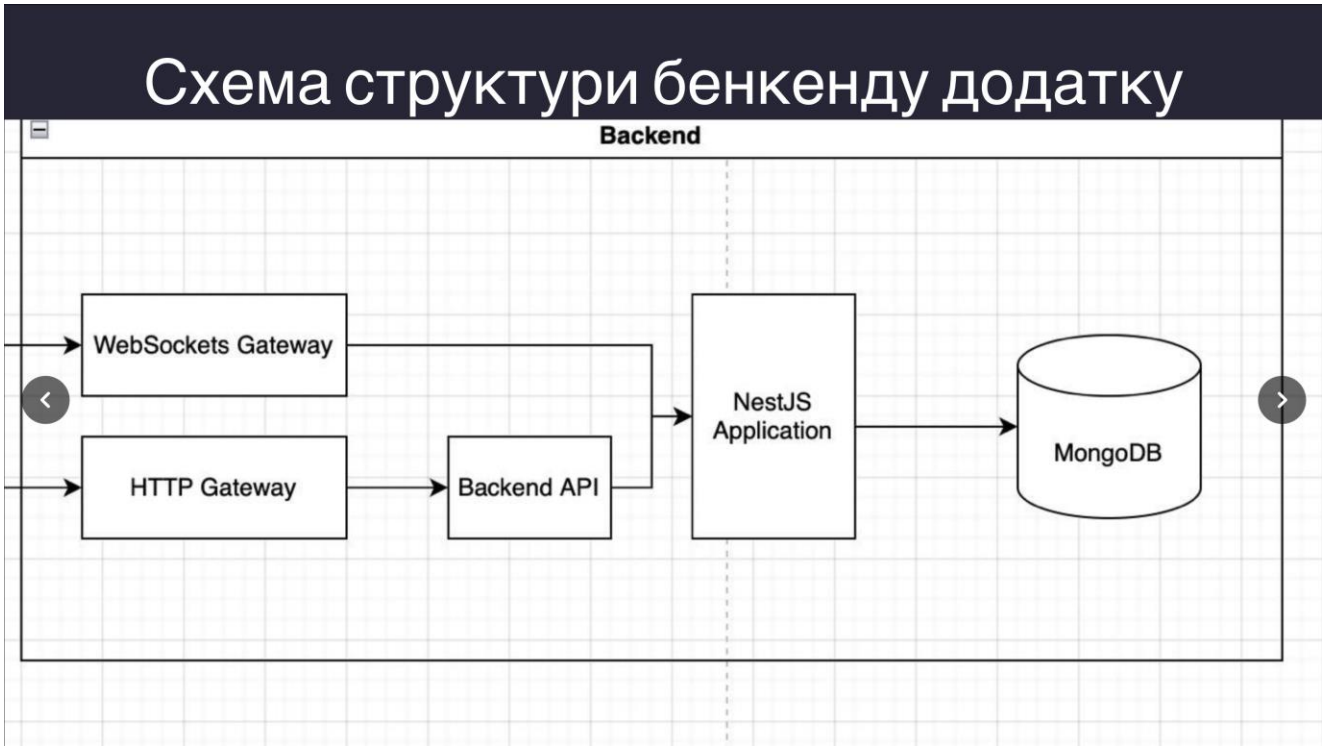
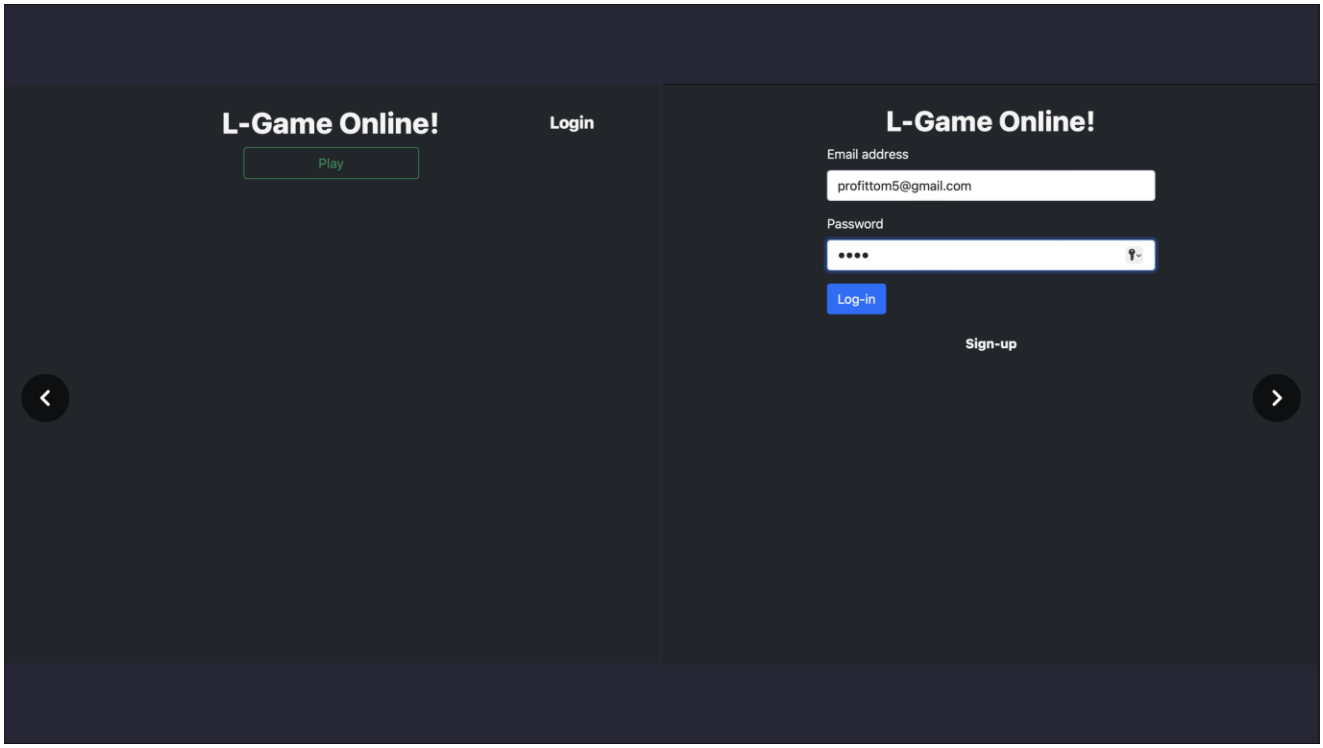
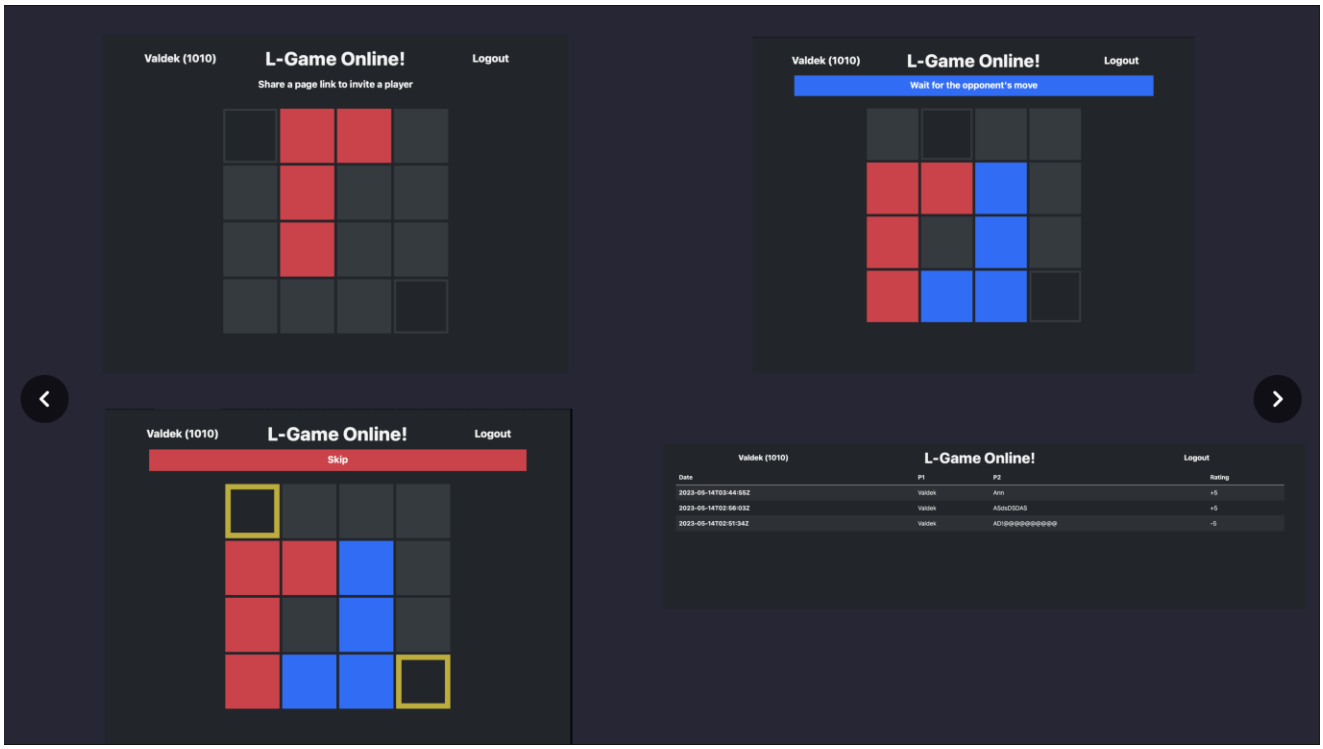


Схема структури бенкенду додатку




L-Game Online!
Login
Play
L-Game Online!
Email address
profittom5@gmail.com
Password
Log-in
Sign-up


Valdek (1010)
L-Game Online!
Logout
Share a page link to invite a player
Valdek (1010)
L-Game Online!
Logout
Wait for the opponent's move
Valdek (1010)
L-Game Online!
Logout
Skip
Valdek (1010)
L-Game Online!
Logout
Date
P1
P2
Rating

ДОДАТОК Г
(обов’язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

Введення

Робота виконується в рамках проекту розробки веб сервісу для онлайн-гри у головоломку «L-game» на базі JS та Node.js. Технічне завдання розроблено у відповідності до стандарту ГОСТ 19.201–78.

1 Підстава для розробки

Підставою для розробки є «Завдання на кваліфікаційну роботу», затверджене завідувачем кафедри інженерії програмного забезпечення. Найменування розробки: Веб-сервіс для онлайн-гри у головоломку «L-game» на базі JS та Node.js.

2 Призначення розробки

2.1 Функціональне призначення

Функціональним призначенням додатку є клієнтська частина для гри у головоломку та відображення результатів, та серверна частина для збереження даних та координацію онлайн гри.

2.2 Експлуатаційне призначення

Програма повинна експлуатуватися на будь-яких пристроях, на яких є браузер. Кінцевим користувачем додатку може виступати будь-яка особа.

3 Вимоги до програми

3.1 Вимоги до функціональних характеристик

Для користувача:

- авторизація та створення акаунту;
- підключення до створених кімнат гри;
- створення кімнат для гри;
- підключення до кімнати за посиланням;
- онлайн гра у головоломку «Л-гра» з іншим користувачем;
- збереження результатів;
- відображення результатів користувача;
- відображення загальної статистики гравців;

3.2 Вимоги до надійності

Веб-сервіс повинен забезпечувати такі вимоги до надійності:

- обробляти невірні дії користувача і попереджати його про можливі наслідки;
- перевіряти авторизованісь запитів користувача;
- шифрувати чутливі дані користувачів;
- захищатись від атак з ціллю нашкодити роботі ПЗ.

3.3 Умови експлуатації та вимоги до технічних засобів

Веб-сервіс повинен працювати на всіх пристроях, які мають браузер та стабільний доступ до мережі «Інтернет»: смартфонах, планшетах та комп'ютерах. Браузер може бути будь-яким, але відповідати тех-вимогам.

Для роботи платформи без збоїв, потрібно, щоб пристрій, на якому вона запускається задовольняв наступні мінімальні вимоги:

- 500Мб оперативної пам'яті;
- 2-ядерний процесор;
- доступ до мережі «Інтернет» зі швидкістю мінімум 5 Мбіт/с.

3.4 Вимоги до інформаційної та програмної сумісності

Для розробки веб-застосунку має використовуватись фронтенд JavaScript, бекенд NodeJS, та база даних NoSQL.

3.5 Спеціальні вимоги

Програма повинна мати зручний та приємний інтерфейс користувача.

4 Вимоги до програмної документації

У момент здачі проекту замовнику надається наступний набір документів:

- текст програми;
- опис програми;
- технічне завдання;
- керівництво користувача.

5 Стадії та етапи розробки

Стадії та етапи розробки веб сервісу для онлайн-гри у головоломку «L-game» на базі JS та Node.js А.1.

Таблиця А.1 – Стадії та етапи розробки проекту

Стадія розробки	Етапи робіт	Зміст робіт
Технічне завдання 01.01.23 – 30.01.23	Обґрунтування необхідності розробки програми	Коротка характеристика програмного забезпечення; підстава і призначення розробки; вимоги до програмної системи і документація; стадії і етапи розробки програми; порядок контролю і приймання
Ескізний проект 31.01.23 – 25.02.23	Розробка ескізного проекту	Попередня розробка структури вхідних і вихідних даних; уточнення середовища програмування; розробка і опис загальної алгоритмічної структури
Технічний проект 25.02.23 – 18.03.23	Розробка технічного проекту	Уточнення структури вхідних і вихідних даних; розробка докладного алгоритму; розробка структури програми
Робочий проект 18.03.23 – 14.04.23	Розробка програмного забезпечення	Реалізація програмного забезпечення; відлагодження; проведення попереднього тестування
Розробка програмної документації 14.04.23 – 20.04.23	Розробка документації до програмного забезпечення	Розробка необхідної документації, передбаченої технічним завданням
Тестування системи 20.04.23 – 29.04.23	Проведення тестування програмного забезпечення	Розробка методики тестування; проведення основних тестів; коректування програмного забезпечення
Впровадження	Підготовка і передача програми	Підготовка і розгортання програмного забезпечення
Розробка програмної документації 16.04.23 – 22.04.23	Розробка документації до програмного забезпечення	Розробка необхідної документації, передбаченої технічним завданням

6 Порядок контролю та приймання

Контроль здійснюється кінцевими користувачами системи, підключеними на етапі тестування додатку.

ДОДАТОК Д
(обов’язковий)

КОД (ЛІСТИНГ) ПРОГРАМИ

Game-state.js

```
class GameState {
    elem = null

    players = {}
    playersPositions = {}

    playerId = null

    status = 'pending' // 'pending' | 'started' | 'end'
    turn = null // playerId
    stage = null // 1 | 2

    availablePlayersColors = null

    constructor() {
        this.availablePlayersColors = ['bg-primary', 'bg-danger']
        this.render()
    }

    isActive() { return this.status == 'started' }
    isYourTurn() { return this.playerId == this.turn }

    render() {
        this.elem = createDiv({
            id: 'game-state'
        })
        this.elem.addClass('game-state')
        this.elem.addClass('d-flex')
        this.elem.addClass('text-white')
        this.elem.addClass('justify-content-center')
        this.elem.addClass('mb-4')
        this.elem.addClass('container-fluid')
    }


    start() {
        this.status = 'started'
        this.setTurn(Object.values(this.players)[0].id)
        this.stage = 1

        this.updateWaitLabel()
        console.log('Game started')
    }

    finish() {
        this.status = 'end'
        console.log('Game finished')
    }

    setTurn(id) {
        this.turn = id
        console.log(`Next turn for Player${id}`)
    }
    nextStage(onSkip, onSecondStage) {
        this.stage = 2

        if(this.turn === this.playerId) {
            this.showButton('Skip', () => {
```

```
            this.nextTurn()
            if(onSkip) onSkip()
        })
    }
}
nextTurn() {
    const playersIds = Object.values(this.players).map(({ id }) => id)

    if(playersIds.indexOf(this.turn) === 0) this.setTurn(playersIds[1])
    else this.setTurn(playersIds[0])
    this.stage = 1

    this.updateWaitLabel()

    Object.values(this.players).map(({ color }) => this.elem.toggleClass(color))
}

updateWaitLabel() {
    if(this.turn !== this.playerId)
        this.showMessage('Wait for the opponent\'s move')
    else
        this.showMessage('Make a move')
}


hideMessage() {
    if(this.message) {
        this.message.popover('hide')
        this.message.remove()
        this.message = null
    }
    this.elem.empty()
}

updateMessage(elem, message) {
    this.hideMessage()

    this.message = elem
    this.message.text(message)
    this.message.addClass('mt-2')
    // this.message.addClass('text-dark')
    this.elem.append(this.message)
}

showButton(message, onClick) {
    this.updateMessage(
        createClickableH4({}, onClick),
        message
    )
}
showPopupButton(message, popupContent, onClick) {
    this.updateMessage(
        createClickablePopover({}, popupContent, onClick)
            .addClass('h5'),
        message
    )
}
showMessage(message) {
    this.updateMessage(
        createH4({}),
        message
```

```
        )
    }

    addPlayer(playerId) {
        console.log(`Adding  player <${playerId}>`)
        if(Object.values(this.players).length >= 2) {
            throw new Error('A maximum of 2 players can participate in the game')
        }

        const playerColor = this.availablePlayersColors.pop()
        if(!playerColor)
            throw new Error('The Game has no available player colors left')

        console.log(`Creating player <${playerId}>`)
        const player = new Player(playerId, playerColor)
        this.players[player.id] = player

        console.log('Player added')
    }

    applyPosition(playerId, position) {
        const player = this.players[playerId]
        position.tiles.map(tile => tile.setOccupation(player))
        this.playersPositions[player.id] = position
    }

    removePosition(playerId) {
        const position = this.playersPositions[playerId]
        position.tiles.map(tile => tile.setOccupation(null))
        this.playersPositions[playerId] = null
    }

    moveCoin(playerId, position) {
        const coin = position.sourceTile.occupation
        position.sourceTile.setOccupation(null)
        position.targetTile.setOccupation(coin)
    }
}
```

game.js

```
class Game {
    elem = null
    tilesContainer = null
    positionsFactory = null

    state = null

    roomId = $('#game-id').text()
    socket = null

    constructor(gameElem) {
        console.log('Creating Game')
        this.elem = gameElem
        this.tilesContainer = new TilesContainer()
        this.render()
        console.log('Game created')
    }

    render() {
```

```
        console.log('Rendering game')

        this.elem.addClass('container')
        this.elem.addClass('d-flex')
        this.elem.addClass('flex-column')
        this.elem.addClass('align-items-center')

        this.elem.append(this.tilesContainer.elem)

        console.log('Game rendered')
    }


    init() {
        console.log('Initializing game')
        // Init game layout
        this.tilesContainer.createLayout()
        UserInput.createTilesSelectionListeners(this)
        Object.values(this.tilesContainer.tiles).map(tile =>
this.tilesContainer.elem.append(tile.elem))

        this.positionsFactory = new PositionsFactory(this.tilesContainer)
        console.log('Game layout assigned')

        this.state = new GameState()
        this.elem.prepend(this.state.elem)

        this.socket = io();
        this.socket.connect()

        this.state.showMessage('Connecting...')
        this.state.elem.find('h4').addClass('text-dark')

        this.socket.on("connect", async () => {
            this.state.hideMessage()
            this.state.playerId = this.socket.id

            this.socket.on('player-joined', ({ users }) => {
                users.forEach(userId => this.addPlayer(userId))

                if(Object.keys(this.state.players).length < 2) {
                    this.state.showPopupButton(`Share a page link to invite a player`,
'Link copied to clipboard', event => navigator.clipboard.writeText(window.location.href))
                }
            })

            this.socket.on('leave', ({ room, id }) => {
                console.log('Leave')
                if(this.state.isActive()) {
                    this.finish('You won! Your opponent has left the game')
                    LGameAPI.finishMatch(this.roomId)
                }
            })

            this.socket.on('change-position', ({ playerId, position: rawPosition }) => {
                if(playerId != this.state.playerId) {
                    console.log('Recived update')
                    if(rawPosition.__typename === LPositionSelection.name) {
                        const position = new LPositionSelection(
                            rawPosition.tiles.map(tile =>
this.tilesContainer.tiles[tile.key]),
```

```
                            playerId
                        )
                        position.hide()
                        this.updatePosition(playerId, position)
                    } else if(rawPosition.__typename === CoinPositionSelection.name) {
                        const position = new CoinPositionSelection(
                            this.tilesContainer.tiles[rawPosition.sourceTile.key],
                            playerId
                        )
                        position.hide()
                        position.targetTile =
this.tilesContainer.tiles[rawPosition.targetTile.key]
                        this.updatePosition(playerId, position)
                    }
                }
            })

            this.socket.on('skip', ({ playerId, position: rawPosition }) => {
                if(playerId != this.state.playerId) {
                    console.log('Recived skip')
                    this.skipStage2()
                }
            })

            await this.socket.emit('join-room', { roomId: this.roomId });

            console.log('Game initialized')
        });
    }

    start() {
        this.state.hideMessage()
        this.state.start()
        this.state.elem.addClass(this.state.players[this.state.turn].color)
    }

    createPlayerInitialPosition(playerId) {
        // For testing
        const position1 = this.positionsFactory.createLPosition(2, 4, 'HB', playerId)
        const position2 = this.positionsFactory.createLPosition(4, 1, 'BH', playerId)

        // const position1 = this.positionsFactory.createLPosition(1, 3, 'VT', playerId)
        // const position2 = this.positionsFactory.createLPosition(4, 2, 'BV', playerId)
        if(position1.isValid())
            return position1
        else
            return position2
    }

    addPlayer(playerId) {
        if(!Object.keys(this.state.players).find(id => id === playerId)) {
            if(playerId == this.state.playerId)
                LGameAPI.joinMatch(this.roomId)

            this.state.addPlayer(playerId)

            const initialPosition = this.createPlayerInitialPosition(playerId)
            this.state.applyPosition(playerId, initialPosition)

            if(Object.values(this.state.players).length >= 2)
                this.start()
```

```
        }
    }

    updatePosition(playerId, position) {
        if(this.state.stage === 1) {
            this.state.removePosition(playerId)
            this.state.applyPosition(playerId, position)
            this.state.nextStage(
                () => {
                    this.socket.emit('skip', { roomId: this.roomId, playerId:
this.state.playerId });
                    this.elem.toggleClass('stage2')
                }
            )
        } else {
            this.state.moveCoin(playerId, position)
            setTimeout(() => position.hide(), 1000)
            this.state.nextTurn()

            this.checkForWinners()
        }
        this.elem.toggleClass('stage2')
    }

    changePosition(playerId, position) {
        if(this.state.stage === 1) {
            this.state.removePosition(playerId)
            this.state.applyPosition(playerId, position)
            this.state.nextStage(
                () => {
                    this.socket.emit('skip', { roomId: this.roomId, playerId:
this.state.playerId })
                    this.checkForWinners()
                }
            )
        } else {
            this.state.moveCoin(playerId, position)
            this.state.nextTurn()

            this.checkForWinners()
        }
        this.elem.toggleClass('stage2')

        this.socket.emit('change-position', { roomId: this.roomId, playerId, position })
    }

    skipStage2() {
        this.state.nextTurn()
        this.elem.toggleClass('stage2')
    }

    checkForWinners() {
        const playersWithAvailableMoves = Object.keys(this.state.players).filter(playerId
=> this.hasAvailableMoves(playerId))
        console.log("Players with available moves: ", playersWithAvailableMoves)
        if(playersWithAvailableMoves.length < 2) {
            const winner = playersWithAvailableMoves[0]
            if(winner == this.state.playerId) {
                this.finish('You won!')
                LGameAPI.finishMatch(this.roomId)
            } else
```

```
                this.finish('You lost :(')
            }
        }
        hasAvailableMoves(playerId) {
            const variants = ['BV', 'TV', 'VB', 'VT', 'BH', 'TH', 'HB', 'HT']
            return Object.values(this.tilesContainer.tiles).some(
                tile => variants.some(
                    variant => this.positionsFactory
                        .createLPosition(tile.row, tile.col, variant, playerId)
                        .isValidMoveFrom(this.state.playersPositions[playerId])
                )
            )
        }

        finish(message) {
            this.socket.disconnect()
            // this.tilesContainer.elem.remove()
            this.state.showMessage(message)
            this.state.finish()
        }
}

const main = async () => {
    const data = await LGameAPI.getUser()
    $('#nickname').text(`${data.nickname} (${data.rating})`)

    console.log('Loaded')
    const gameElem = $('#game')
    const game = new Game(gameElem)
    game.init()
}

$(() => { main().catch(e => { throw e }) })
```

Position.js

```
class Position {
    playerId = null
    constructor(playerId) {
        this.playerId = playerId
    }
}

class CoinPositionSelection extends Position {
    __typename = CoinPositionSelection.name
    isActive = true

    sourceTile = null
    targetTile = null
    constructor(sourceTile, playerId) {
        super(playerId)
        this.sourceTile = sourceTile
    }

    isValid() {
        if(!this.sourceTile || !this.targetTile) return false
        return this.sourceTile.occupation instanceof Coin && this.targetTile.occupation
=== null
    }
```

```
    render() {
        if(this.sourceTile)
            this.sourceTile.elem.addClass('selected-y')

        if(this.targetTile) {
            if(this.isValid())
                this.targetTile.elem.addClass('selected')
            else
                this.targetTile.elem.addClass('selected-r')
        }
    }

    hide() {
        if(this.sourceTile)
            this.sourceTile.elem.removeClass('selected-y')

        if(this.targetTile) {
            this.targetTile.elem.removeClass('selected')
            this.targetTile.elem.removeClass('selected-r')
        }
    }

    setTarget(tile) {
        this.hide()
        this.targetTile = tile
        this.render()
    }

    finish() {
        this.isActive = false
    }
}

class LPosition extends Position {
    __typename = LPosition.name

    tiles = null
    constructor(tiles, playerId) {
        super(playerId)
        this.tiles = tiles.filter(Boolean)
    }

    isOverloaded() {
        return this.tiles.length > 4;
    }

    isFull() {
        return this.tiles.length >= 4;
    }

    isEmpty() {
        return this.tiles.length === 0
    }

    isFreeSpace() {
        return this.tiles.length >= 0 && this.tiles.length < 4
    }

    isValidShape() {
        const rows = this.tiles.map(tile => tile.row)
        const cols = this.tiles.map(tile => tile.col)
```

```
        const rowsU = new Set(rows)
        const colsU = new Set(cols)
        if(rowsU.size === 2 && colsU.size === 3) {

            const count = {}
            rows.map(row => count[row] = (count[row] || 0) + 1)

            if(!Object.values(count).find((c) => c === 3)) return false
            else return true
        } else if(rowsU.size === 3 && colsU.size === 2) {

            const count = {}
            cols.map(col => count[col] = (count[col] || 0) + 1)

            if(!Object.values(count).find((c) => c === 3)) return false
            else return true
        } else {
            return false
        }

    }

    isOutOfBorders() {
        const rows = this.tiles.map(tile => tile.row)
        const cols = this.tiles.map(tile => tile.col)
        return rows.some(row => row < 1 || row > 4) || cols.some(col => col < 1 || col >
4)
    }

    isEqual(lPosition) {
        return this.tiles
            .map((tile, i, tiles) => !!lPosition.tiles.find(t => t.isEqual(tile)))
            .reduce((prev, curr) => prev && curr, true)

    }

    isValid() {
        if(!(this.isFull() && !this.isOverloaded() && !this.isOutOfBorders()))
            return false

        if(!this.isValidShape())
            return false

        const validTilesPositions = this.tiles.map(
            tile => {
                if(tile.occupation instanceof Coin) {
                    return false
                } else if(tile.occupation === null) {
                    return true
                } else if(tile.occupation instanceof Player) {
                    const occupant = tile.occupation
                    return occupant.id === this.playerId
                }
            }
        )
        return validTilesPositions.reduce((prev, curr) => prev && curr, true)
    }

    isValidMoveFrom(originLPosition) {
        return this.isValid() && !this.isEqual(originLPosition)
    }
```

```
    toggleHint() {
        this.tiles.map(tile => tile.toggleHint())
    }
}

class LPositionSelection extends LPosition {
    __typename = LPositionSelection.name
    isActive = true

    constructor(tiles, playerId) {
        super(tiles, playerId)
        this.render()
    }

    finish() {
        this.isActive = false
    }

    addTile(tile) {
        this.hide()
        this.tiles.push(tile)
        this.render()
    }

    removeTile(tile) {
        const index = this.tiles.findIndex((t => t.isEqual(tile)))
        if(index > -1) {
            this.hide()
            this.tiles.splice(index, 1);
            this.render()
        }
    }

    render() {
        this.tiles.map(tile => {
            if(this.isValid())
                tile.elem.addClass('selected')
            else
                tile.elem.addClass('selected-r')
        })
    }

    hide() {
        this.tiles.map(tile => {
            tile.elem.removeClass('selected')
            tile.elem.removeClass('selected-r')
        })
    }
}

class PositionsFactory {

    constructor(tilesContainer) {
        this.tilesContainer = tilesContainer
    }

    /**
     *
     * @param {*} headr - head tile row number
     * @param {*} headc - head tile column number
```

```
     * @param {*} type  - type of position:
     *  'VB':          'BV':        'VT':         'TV':         'TH':              'BH':
'HT':            'HB':
     *      *               *           * *             * *            *
*
     *      *               *           *                 *            * * *           *
* *           * * *           * * *
     *      * *           * *           *                 *                           *
*
     */
    createLPosition(headr, headc, type, playerId) {
        if(type == 'VB')
            return new LPosition([
                this.tilesContainer.tiles[`${headr}-${headc}`],
                this.tilesContainer.tiles[`${headr}-${headc-1}`],
                this.tilesContainer.tiles[`${headr-1}-${headc-1}`],
                this.tilesContainer.tiles[`${headr-2}-${headc-1}`],
            ], playerId)
        else if(type == 'BV')
            return new LPosition([
                this.tilesContainer.tiles[`${headr}-${headc}`],
                this.tilesContainer.tiles[`${headr}-${headc+1}`],
                this.tilesContainer.tiles[`${headr-1}-${headc+1}`],
                this.tilesContainer.tiles[`${headr-2}-${headc+1}`],
            ], playerId)
        else if(type == 'VT')
            return new LPosition([
                this.tilesContainer.tiles[`${headr}-${headc}`],
                this.tilesContainer.tiles[`${headr}-${headc-1}`],
                this.tilesContainer.tiles[`${headr+1}-${headc-1}`],
                this.tilesContainer.tiles[`${headr+2}-${headc-1}`],
            ], playerId)
        else if(type == 'TV')
            return new LPosition([
                this.tilesContainer.tiles[`${headr}-${headc}`],
                this.tilesContainer.tiles[`${headr}-${headc+1}`],
                this.tilesContainer.tiles[`${headr+1}-${headc+1}`],
                this.tilesContainer.tiles[`${headr+2}-${headc+1}`],
            ], playerId)
        else if(type == 'TH')
            return new LPosition([
                this.tilesContainer.tiles[`${headr}-${headc}`],
                this.tilesContainer.tiles[`${headr+1}-${headc}`],
                this.tilesContainer.tiles[`${headr+1}-${headc+1}`],
                this.tilesContainer.tiles[`${headr+1}-${headc+2}`],
            ], playerId)
        else if(type == 'BH')
            return new LPosition([
                this.tilesContainer.tiles[`${headr}-${headc}`],
                this.tilesContainer.tiles[`${headr-1}-${headc}`],
                this.tilesContainer.tiles[`${headr-1}-${headc+1}`],
                this.tilesContainer.tiles[`${headr-1}-${headc+2}`],
            ], playerId)
        else if(type == 'HT')
            return new LPosition([
                this.tilesContainer.tiles[`${headr}-${headc}`],
                this.tilesContainer.tiles[`${headr+1}-${headc}`],
                this.tilesContainer.tiles[`${headr+1}-${headc-1}`],
                this.tilesContainer.tiles[`${headr+1}-${headc-2}`],
            ], playerId)
        else if(type == 'HB')
```

```
            return new LPosition([
                this.tilesContainer.tiles[`${headr}-${headc}`],
                this.tilesContainer.tiles[`${headr-1}-${headc}`],
                this.tilesContainer.tiles[`${headr-1}-${headc-1}`],
                this.tilesContainer.tiles[`${headr-1}-${headc-2}`],
            ], playerId)
        else throw new Error('Invalid position type')
    }

}
```

tile.js

```
class Tile {
    elem = null

    row = null
    col = null
    key = null

    occupation = null // Player | Coin
    occupationClass = null

    constructor(row, col, occupation) {
        this.row = row
        this.col = col
        this.key = `${row}-${col}`

        if(occupation !== undefined)
            this.occupation = occupation

        this.render()
    }

    setOccupation(occupant) {
        this.removeOccupationClass()
        this.occupation = occupant
        this.addOccupationClass()
    }

    render() {
        const attr = {
            id: `tile-${this.row}-${this.col}`
        }

        this.elem = createDiv(attr)

        // const hint = createDiv({})
        // hint.addClass('c')
        // this.elem.append(hint)

        this.elem.data('pos', { r: this.row, c: this.col })
        this.elem.addClass('tile')
        this.elem.addClass('bg-tile-e')
        // this.elem.text(`${this.row}-${this.col}`)
        this.addOccupationClass()
    }

    addOccupationClass() {
        let label = 'bg-white'
```

```
        if(this.occupation === null) {
            label = 'bg-tile-e'
        } else if(this.occupation instanceof Coin) {
            label = 'bg-dark selected-y'
        } else if(this.occupation instanceof Player) {
            const player = this.occupation
            label = player.color
        }

        this.elem.addClass(label)
        this.occupationClass = label
    }

    removeOccupationClass() {
        this.elem.removeClass(this.occupationClass)
    }

    isEqual(tile) {
        return this.row === tile.row && this.col === tile.col
    }

    toggleHint() {
        this.elem.toggleClass('hint')
    }
}
```

Tiles-container.js

```
class TilesContainer {
    elem = null

    tiles = null

    currentSelection = null

    constructor() {
        this.render()
    }

    render() {
        this.elem = createDiv({
            id: 'game-tiles-container'
        })
        this.elem.addClass('game-tiles-container')
    }

    // Layout init
    createLayout() {
        console.log('Creating game layout')
        // For now, only default layout enabled
        this.tiles = {
            // Row 1
            '1-1': new Tile(1, 1, new Coin()),
            '1-2': new Tile(1, 2),
            '1-3': new Tile(1, 3),
            '1-4': new Tile(1, 4),

            // Row 2
            '2-1': new Tile(2, 1),
```

```
            '2-2': new Tile(2, 2),
            '2-3': new Tile(2, 3),
            '2-4': new Tile(2, 4),

            // Row 3
            '3-1': new Tile(3, 1),
            '3-2': new Tile(3, 2),
            '3-3': new Tile(3, 3),
            '3-4': new Tile(3, 4),

            // Row 4
            '4-1': new Tile(4, 1),
            '4-2': new Tile(4, 2),
            '4-3': new Tile(4, 3),
            '4-4': new Tile(4, 4, new Coin()),
        }
    }

    clearSelection() {
        this.currentSelection.hide()
        this.currentSelection = null
    }
}
```

User-input.js

```
class UserInput{
      game.tilesContainer.elem.mouseleave(function(){
         if(game.tilesContainer.currentSelection)
            game.tilesContainer.clearSelection()
      });
      game.elem.mouseup(function(){
         if(game.tilesContainer.currentSelection){
            if(game.tilesContainer.currentSelection.isValid()){
               if(game.state.stage === 1) {
                  if(!(
                     game.tilesContainer.currentSelection
                        .isEqual(game.state.playersPositions[game.state.turn])
                  ))
                     game.changePosition(game.state.turn,
game.tilesContainer.currentSelection)
               } else {
                  if(!(
                     game.tilesContainer.currentSelection.targetTile
                        .isEqual(game.tilesContainer.currentSelection.sourceTile)
                  ))
                     game.changePosition(game.state.turn,
game.tilesContainer.currentSelection)
               }

            }

            game.tilesContainer.clearSelection()
         }
```

```
        });
    }
    // static createSkipStage2ButtonListeners(state, cb) {
    //     state.message.click(function() {
    //         state.nextTurn()
    //         if(cb) cb()
    //     })
    // }
}
```

Api.js

```
class Auth {
    static getToken() {
        if(!localStorage.getItem("token"))
            document.location.href = '/login'
        else
            Auth.enableLogout()
        return localStorage.getItem("token")
    }

    static setToken(token) {
        localStorage.setItem("token", token)
        Auth.enableLogout()
    }

    static rejectToken() {
        localStorage.setItem("token", null)
        $('#login-nav-btn').text('Login')
    }

    static enableLogout() {
        $('#login-nav-btn').text('Logout').on('click', e => {
            e.preventDefault()
            Auth.rejectToken()
            document.location.href = '/login'
        })
    }
}

class LGameAPI {
    static handleError(xhr) {
        const { error, message, statusCode } = xhr.responseJSON

        if(statusCode == 401) {
            Auth.rejectToken();
        } else {
            $('#error-message').text(`Server Error (${error}), status: ${statusCode} -
${message}`)
            const errorToast = document.getElementById('errorToast')
            if(errorToast) {
                const toastBootstrap = bootstrap.Toast.getOrCreateInstance(errorToast)
                toastBootstrap
                toastBootstrap.show()
            } else {
                console.log(xhr.responseJSON)
            }
```

```
        }
    }

    static async get(path, errorCb) {
        return new Promise((res, rej) => {
            $.ajax({
                type: "GET",
                beforeSend: function(request) {
                  request.setRequestHeader("Authorization", 'Bearer ' + Auth.getToken());
                },
                url: `${HOST}${path}`,
                processData: false,
                success: res,
                error: errorCb //xhr => { LGameAPI.handleError(xhr);
rej(xhr.responseJSON.message) }
            });
        })
    }
    static async post(path, data) {

    }

    static async getUser() {
        return new Promise((res, rej) => {
            $.ajax({
                type: "GET",
                beforeSend: function(request) {
                  request.setRequestHeader("Authorization", 'Bearer ' + Auth.getToken());
                },
                url: `${HOST}/users/me`,
                processData: false,
                success: res,
                error: xhr => { LGameAPI.handleError(xhr); rej(xhr.responseJSON.message) }
            });
        })
    }

    static async findMatch() {
        return new Promise((res, rej) => {
            $.ajax({
                type: "GET",
                beforeSend: function(request) {
                  request.setRequestHeader("Authorization", 'Bearer ' + Auth.getToken());
                },
                url: `${HOST}/match/random`,
                processData: false,
                success: res,
                error: xhr => LGameAPI.handleError(xhr)
            });
        })
    }


    static async joinMatch(roomId) {
        return new Promise((res, rej) => {
            $.ajax({
                type: "POST",
                data: JSON.stringify({ roomId }),
                contentType: 'application/json',
                beforeSend: function(request) {
                  request.setRequestHeader("Authorization", 'Bearer ' + Auth.getToken());
```

```
            },
            url: `${HOST}/match/join`,
            processData: true,
            success: res,
            error: xhr => LGameAPI.handleError(xhr)
        });
    })
}

static async finishMatch(roomId) {
    return new Promise((res, rej) => {
        $.ajax({
            type: "POST",
            data: JSON.stringify({ roomId }),
            contentType: 'application/json',
            beforeSend: function(request) {
              request.setRequestHeader("Authorization", 'Bearer ' + Auth.getToken());
            },
            url: `${HOST}/match/finish`,
            processData: true,
            success: res,
            error: xhr => LGameAPI.handleError(xhr)
        });
    })
}

static async getStats(userId) {
    return new Promise((res, rej) => {
        $.ajax({
            type: "GET",
            beforeSend: function(request) {
              request.setRequestHeader("Authorization", 'Bearer ' + Auth.getToken());
            },
            url: `${HOST}/match`,
            processData: false,
            success: res,
            error: xhr => LGameAPI.handleError(xhr)
        });
    })
}

static async getTopUsers() {
    return new Promise((res, rej) => {
        $.ajax({
            type: "GET",
            beforeSend: function(request) {
              request.setRequestHeader("Authorization", 'Bearer ' + Auth.getToken());
            },
            url: `${HOST}/users/top`,
            processData: false,
            success: res,
            error: xhr => LGameAPI.handleError(xhr)
        });
    })
}
}
```

App.controller.ts

```
import { Controller, Get, HttpServer, Inject, Param, Redirect, Render, Req } from
'@nestjs/common';
import { HttpAdapterHost } from '@nestjs/core';
import { randomUUID } from 'crypto';
import { v4 as uuidv4 } from 'uuid';

@Controller()
export class AppController {
  @Inject(HttpAdapterHost)
  httpServerRef: HttpAdapterHost

  @Get('stats')
  @Render('stats')
  stats() {
    return {};
  }

  @Get('login')
  @Render('login')
  login() {
    return {};
  }

  @Get('signup')
  @Render('register')
  register() {
    return {};
  }

  @Get()
  @Render('menu')
  menu() {
    return {};
  }

  @Get('play')
  @Redirect('/', 302)
  async play(@Req() req) {
    return {
      url: `${req.protocol + '://' + req.get('host')/*await
global['app'].getUrl()*/}/room/${uuidv4()}`
    }
  }


  @Get('room/:roomId')
  @Render('room')
  room(@Param('roomId') roomId: string) {
    return {
      roomId
    };
  }
}
```

Users.controller.ts

```
import { Body, Controller, Post, HttpCode, HttpStatus, Request, Get, UseGuards } from
'@nestjs/common';
import { AuthGuard } from 'src/auth/auth.guard';
import { UsersService } from './users.service';
```

```
@Controller('users')
export class UsersController {
    constructor(
        private usersService: UsersService
    ) {}

    @UseGuards(AuthGuard)
    @HttpCode(HttpStatus.OK)
    @Get('me')
    getMe(@Request() req) {
        console.log(req.user)
        return this.usersService.findOne(req.user.email);
    }

    @HttpCode(HttpStatus.OK)
    @Get('top')
    getTop(@Request() req) {
        return this.usersService.findTop5();
    }
}
```

Matches.controller.ts

```
import { Body, Controller, Post, HttpCode, HttpStatus, Request, Get, UseGuards } from
'@nestjs/common';
import { AuthGuard } from 'src/auth/auth.guard';
import { EMatchStatus } from 'src/schemas/match.schema';
import { UsersService } from 'src/users/users.service';
import { MatchesService } from './matches.service';

@Controller('match')
export class MatchesController {
    constructor(
        private matchesService: MatchesService,
        private usersService: UsersService,
    ) {}

    @UseGuards(AuthGuard)
    @HttpCode(HttpStatus.OK)
    @Get('')
    async getMatches(@Request() req) {
        const user = await this.usersService.findOne(req.user.email)
        const matches = await this.matchesService.findByUser(user.id)
        const usersMap = {}; matches.forEach(({ player1, player2 }) => {
            if(player1) usersMap[player1] = null;
            if(player2) usersMap[player2] = null;
        })
        const users = await this.usersService.findByIds(Object.keys(usersMap))
        users.forEach(user => usersMap[user.id] = user)
        return matches.map(match => ({
            ...match.toJSON(),
            player1: usersMap[match.player1],
            player2: usersMap[match.player2],
        }))
    }

    @UseGuards(AuthGuard)
    @HttpCode(HttpStatus.OK)
    @Get('/random')
```

```
    async getRandomOngoingMatch(@Request() req) {
        return this.matchesService.findOnePending()
    }

    @UseGuards(AuthGuard)
    @HttpCode(HttpStatus.OK)
    @Post('/join')
    async connectedToMatch(@Request() req, @Body() body: { roomId: string }) {
        console.log(body)
        const user = await this.usersService.findOne(req.user.email)
        return this.matchesService.join(body.roomId, user.id);
    }

    @UseGuards(AuthGuard)
    @HttpCode(HttpStatus.OK)
    @Post('/finish')
    async finishMatch(@Request() req, @Body() body: { roomId: string }) {
        const user = await this.usersService.findOne(req.user.email)
        const match = await this.matchesService.finish(body.roomId, user.id)
        user.rating += match.winnerRatingChange
        await user.save()
        return user;
    }

}
```

Matches.gateway.ts

```
import {
  MessageBody,
  OnGatewayInit,
  SubscribeMessage,
  WebSocketGateway,
  WebSocketServer,
  ConnectedSocket
} from '@nestjs/websockets';
import { Server, Socket } from 'socket.io';
import { EMatchStatus } from 'src/schemas/match.schema';
import { UsersService } from 'src/users/users.service';
import { MatchesService } from './matches.service';

@WebSocketGateway({
  cors: {
    origin: '*',
  },
})
export class MatchesGateway implements OnGatewayInit<Server> {

    constructor(
        private matchesService: MatchesService
    ) {}

    @WebSocketServer()
    server: Server;

    getRoomUsers(roomId) {
```

```
        const res = this.server.sockets.adapter.rooms.get(roomId)
        if(res) return [...res]
        else return []
    }

    afterInit(server: Server) {
        server.of("/").adapter.on("leave-room", async (room, id) => {
            const match = await this.matchesService.findOne(room)
            if(match) {
                const users = this.getRoomUsers(room)
                console.log(users)
                if(users.length === 0) {
                    match.status = EMatchStatus.FINISHED
                    await match.save()
                }
            }
            server.in(room).emit('leave', { room, id });
        });
    };

    // @SubscribeMessage('leave')
    // async leaveRoom(@MessageBody() data: { room, id }, @ConnectedSocket() socket:
Socket) {
    // }

    @SubscribeMessage('join-room')
    async joinRoom(@MessageBody() data, @ConnectedSocket() socket: Socket) {
        console.log(`${socket.id} joins room ${data.roomId}`)
        let users = this.getRoomUsers(data.roomId)
        if(users.length == 0)
            await this.matchesService.create({ player1: null, player2: null, status:
EMatchStatus.PENDING, winner: null, winnerRatingChange: null, socketRoomId:
data.roomId, isPrivate: false })
        await socket.join(data.roomId);
        users = this.getRoomUsers(data.roomId)
        await this.server.in(data.roomId).emit('player-joined', { users });
    }

    @SubscribeMessage('change-position')
    async changePosition(@MessageBody() { roomId, playerId, position },
@ConnectedSocket() socket: Socket) {
        console.log(`${playerId} made move`)
        await this.server.in(roomId).emit('change-position', { playerId, position });
    }

    @SubscribeMessage('skip')
    async skipMove(@MessageBody() { roomId, playerId }, @ConnectedSocket() socket:
Socket) {
        console.log(`${playerId} made skip`)
        await this.server.in(roomId).emit('skip', { playerId });
    }
}
```

Match, schema.ts

```
import { Prop, Schema, SchemaFactory } from '@nestjs/mongoose';
import { HydratedDocument } from 'mongoose';

export type MatchDocument = HydratedDocument<Match>;

export enum EMatchStatus {
  PENDING = 'pending',
  STARTED = 'started',
  FINISHED = 'end',
}

@Schema({ timestamps: true })
export class Match {

  @Prop()
  player1: string

  @Prop()
  player2: string

  //{ enum: [EMatchStatus.FINISHED, EMatchStatus.STARTED, EMatchStatus.PENDING] }
  @Prop()
  status: EMatchStatus

  @Prop()
  winner?: string

  @Prop()
  winnerRatingChange?: number

  @Prop()
  socketRoomId: string

  @Prop({ default: false })
  isPrivate: boolean
}

export const MatchSchema = SchemaFactory.createForClass(Match);
```

User.schema.ts

```
import { Prop, Schema, SchemaFactory } from '@nestjs/mongoose';
import { HydratedDocument } from 'mongoose';

export type UserDocument = HydratedDocument<User>;

@Schema()
export class User {
  @Prop()
  nickname: string;
```

```
  @Prop()
  email: string;

  @Prop()
  password: string;

  @Prop()
  rating: number;

}

export const UserSchema = SchemaFactory.createForClass(User);
```

Login.ejs

```
<!DOCTYPE html>
<html class="bg-dark">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <!-- Bootstrap CSS -->
  <link
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
    rel="stylesheet"
    integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
    crossorigin="anonymous"
  />
  <link rel="stylesheet" href="/room.css">
  <title>L-Game Online Log-in</title>
</head>

<body class="bg-dark">
  <div class="container-fluid">

    <div class="text-center">
      <div id="menu-title" class="row align-items-center mt-4">
        <a class="col h1" href="/">L-Game Online!</a>
      </div>
    </div>

    <div class="row justify-content-center">
      <form id="login" class="">
        <div class="mb-3">
          <label for="exampleInputEmail1" class="form-label">Email address</label>
          <input name="email" type="email" class="form-control" id="exampleInputEmail1"
aria-describedby="emailHelp">
        </div>
        <div class="mb-3">
          <label for="exampleInputPassword1" class="form-label">Password</label>
          <input name="password" type="password" class="form-control"
id="exampleInputPassword1">
        </div>
        <div id="login-btn" class="btn btn-primary">Log-in</div>
      </form>
    </div>

    <div class="text-center">
      <div id="menu-title" class="row align-items-center mt-4">
```

```
        <a href="/signup">Sign-up</a>
      </div>
    </div>

  </div>


  <div class="toast-container position-fixed bottom-0 end-0 p-3">
    <div id="errorToast" class="toast text-bg-danger" role="alert" aria-live="assertive"
aria-atomic="true">
      <div class="toast-header">
        <img src="..." class="rounded me-2" alt="...">
        <strong class="me-auto">Server</strong>
        <button type="button" class="btn-close btn-close-white" data-bs-dismiss="toast"
aria-label="Close"></button>
      </div>
      <div id="error-message" class="toast-body">
        User already exist
      </div>
    </div>
  </div>
</body>
<script
  src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
  integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
  crossorigin="anonymous"
></script>
<script src="http://code.jquery.com/jquery-1.11.0.min.js"></script>
<script src="/js/helpers/creds.js"></script>
<script src="/js/api.js"></script>
<script src="/js/login.js"></script>
</html>
```

Menu.ejs

```
<!DOCTYPE html>
<html class="bg-dark">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <!-- Bootstrap CSS -->
  <link
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
    rel="stylesheet"
    integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
    crossorigin="anonymous"
  />
  <link rel="stylesheet" href="/room.css">
  <title>L-Game Online</title>
</head>

<body class="bg-dark">
  <div class="container-fluid text-center">
    <div id="menu-title" class="row align-items-center mt-4">
      <a class="col-4 h5" id="nickname" href="/stats"></a>
      <a class="col-4 h1" href="/">L-Game Online!</a>
      <a class="col-4 h5" id="login-nav-btn" href="/login">Login</a>
    </div>
```

```
    <div id="menu-body" class="mt-5 d-flex justify-content-center">
      <a id="menu-connect-button" class="btn btn-outline-success w-25">
        Play
      </a>
    </div>
    <div class="mt-2 d-flex justify-content-center">
      <a href="/play" id="menu-play-button" class="btn btn-outline-success w-25">
        Create room
      </a>
    </div>

  <div class="toast-container position-fixed bottom-0 end-0 p-3">
    <div id="errorToast" class="toast text-bg-danger" role="alert" aria-live="assertive"
aria-atomic="true">
      <div class="toast-header">
        <img src="..." class="rounded me-2" alt="...">
        <strong class="me-auto">Server</strong>
        <button type="button" class="btn-close btn-close-white" data-bs-dismiss="toast"
aria-label="Close"></button>
      </div>
      <div id="error-message" class="toast-body">
        User already exist
      </div>
    </div>
  </div>
</body>
<script
  src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
  integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
  crossorigin="anonymous"
></script>
<script src="http://code.jquery.com/jquery-1.11.0.min.js"></script>
<script src="/js/helpers/creds.js"></script>
<script src="/js/api.js"></script>
<script src="/js/menu.js"></script>
</html>
```

Register.ejs

```
<!DOCTYPE html>
<html class="bg-dark">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <!-- Bootstrap CSS -->
  <link
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
    rel="stylesheet"
    integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
    crossorigin="anonymous"
  />
  <link rel="stylesheet" href="/room.css">
  <title>L-Game Online</title>
</head>

<body class="bg-dark">
  <div class="container-fluid">

    <div class="text-center">
```

```
      <div id="menu-title" class="row align-items-center mt-4">
        <a class="col h1" href="/">L-Game Online!</a>
      </div>
    </div>

    <div class="row justify-content-center">
      <form id="register">
        <div class="mb-3">
          <label for="exampleInputName" class="form-label">Nickname</label>
          <input name="nickname" class="form-control" id="exampleInputName" aria-
describedby="nickname">
        </div>
        <div class="mb-3">
          <label for="exampleInputEmail1" class="form-label">Email address</label>
          <input name="email" type="email" class="form-control" id="exampleInputEmail1"
aria-describedby="emailHelp">
          <div id="emailHelp" class="form-text">We'll never share your email with anyone
else.</div>
        </div>
        <div class="mb-3">
          <label for="inputPassword5" class="form-label">Password</label>
          <input name="password" type="password" id="inputPassword5" class="form-control"
aria-labelledby="passwordHelpBlock">
          <div id="passwordHelpBlock" class="form-text">
            Your password must be 8-20 characters long, contain letters and numbers, and
must not contain spaces, special characters, or emoji.
          </div>
        </div>
        <div class="mb-3">
          <label for="exampleInputPassword1" class="form-label">Confirm Password</label>
          <input name="password-confirm" type="password" class="form-control"
id="exampleInputPassword1">
        </div>
        <div id="register-btn" class="btn btn-primary">Sign-up</div>
      </form>
    </div>

    <div class="text-center">
      <div id="menu-title" class="row align-items-center mt-4">
        <a href="/login">Log-in</a>
      </div>
    </div>

  </div>


  <div class="toast-container position-fixed bottom-0 end-0 p-3">
    <div id="errorToast" class="toast text-bg-danger" role="alert" aria-live="assertive"
aria-atomic="true">
      <div class="toast-header">
        <img src="..." class="rounded me-2" alt="...">
        <strong class="me-auto">Server</strong>
        <button type="button" class="btn-close btn-close-white" data-bs-dismiss="toast"
aria-label="Close"></button>
      </div>
      <div id="error-message" class="toast-body">
        User already exist
      </div>
    </div>
  </div>
</body>
```

```
<script
  src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
  integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
  crossorigin="anonymous"
></script>
<script src="http://code.jquery.com/jquery-1.11.0.min.js"></script>
<script src="/js/helpers/creds.js"></script>
<script src="/js/register.js"></script>
</html>
```

Room.ejs

```
<!DOCTYPE html>
<html class="bg-dark">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <!-- Bootstrap CSS -->
  <link
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
    rel="stylesheet"
    integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
    crossorigin="anonymous"
  />
  <link rel="stylesheet" href="/room.css">
  <title>L-Game Online</title>
</head>

<body class="bg-dark">
  <div class="container-fluid text-center">
    <div id="menu-title" class="row align-items-center mt-4">
      <a class="col-4 h5" id="nickname" href="/stats"></a>
      <a class="col-4 h1" href="/">L-Game Online!</a>
      <a class="col-4 h5" id="login-nav-btn" href="/login">Login</a>
    </div>
    <div id="game">
      <div id="game-id"><%= roomId %></div>
    </div>
  </div>
</body>
<script
  src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
  integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
  crossorigin="anonymous"
></script>
<script src="http://code.jquery.com/jquery-1.11.0.min.js"></script>
<script src="/js/helpers/creds.js"></script>
<script src="/js/api.js"></script>
<script src="/js/game/tiles-container.js"></script>
<script src="/js/game/user-input.js"></script>
<script src="/js/game/game.js"></script>
<script src="/js/game/game-state.js"></script>
<script src="/js/game/position.js"></script>
<script src="/js/game/coin.js"></script>
<script src="/js/game/player.js"></script>
<script src="/js/game/tile.js"></script>
<script src="/js/helpers/index.js"></script>
<script src="/socket.io/socket.io.js"></script>
</html>
```

```
}
```

ГРАФІЧНА ЧАСТИНА

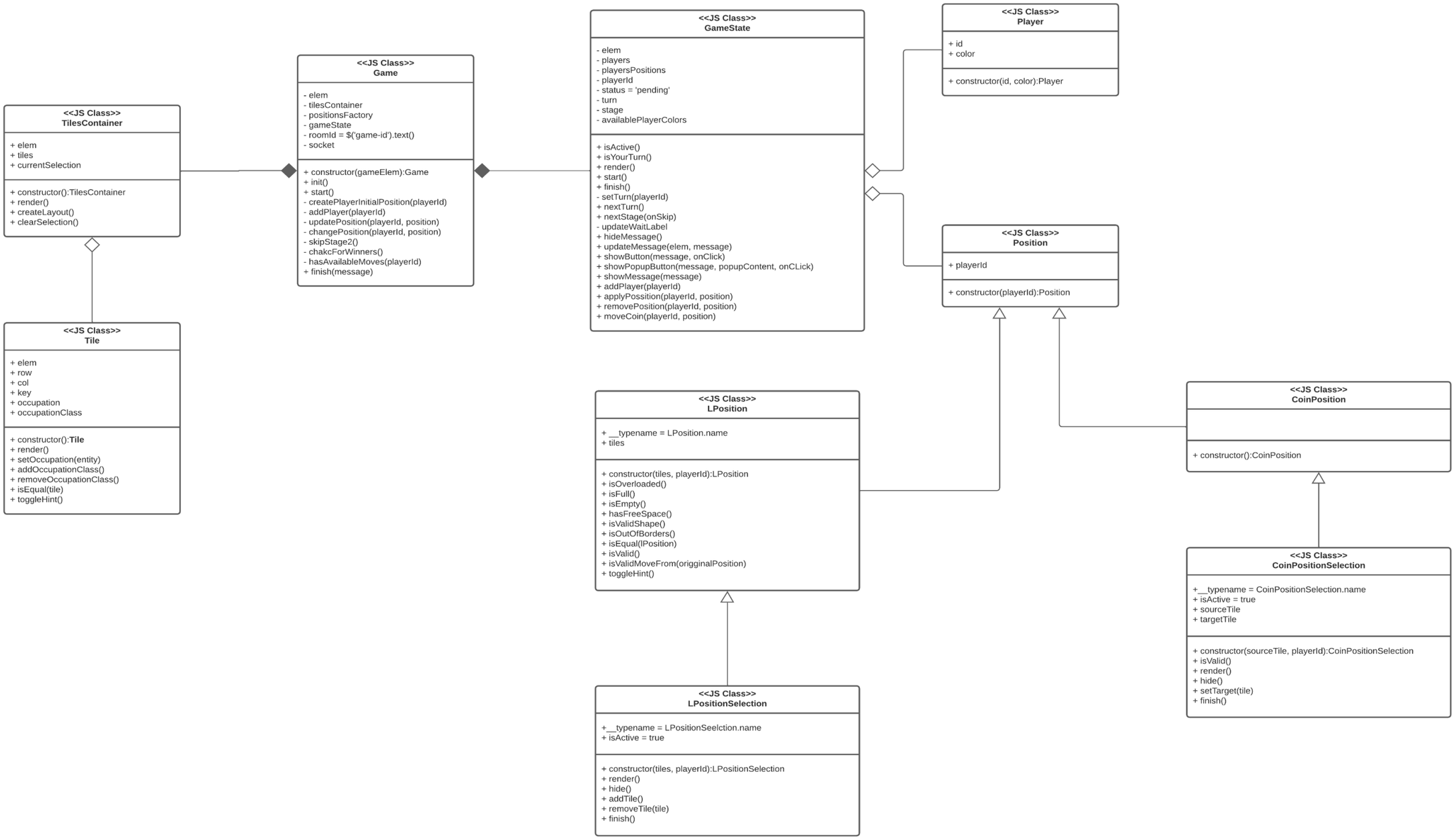


Рисунок 1 – Діаграма класів

					КвРІПЗ.200118.01.01.Е8	
Зм.	Арк.	№докум.	Підпис	Дата	Веб-сервіс для онлайн-гри у головоломку «L-game» на базі JS та Node.js	Літера / Маса / Масштаб
Розробив		Бадьора Я. М		22.05.23		
Керівник		Форкун Ю. В.		22.05.23		Аркуш 1 / Аркушів 3
Консульт.						
Н.Контр.		Праворська Н.І		22.05.23		ХНУ, ІПЗс-20-1
Зав.каф.		Бедратюк Л.П.		22.05.23		

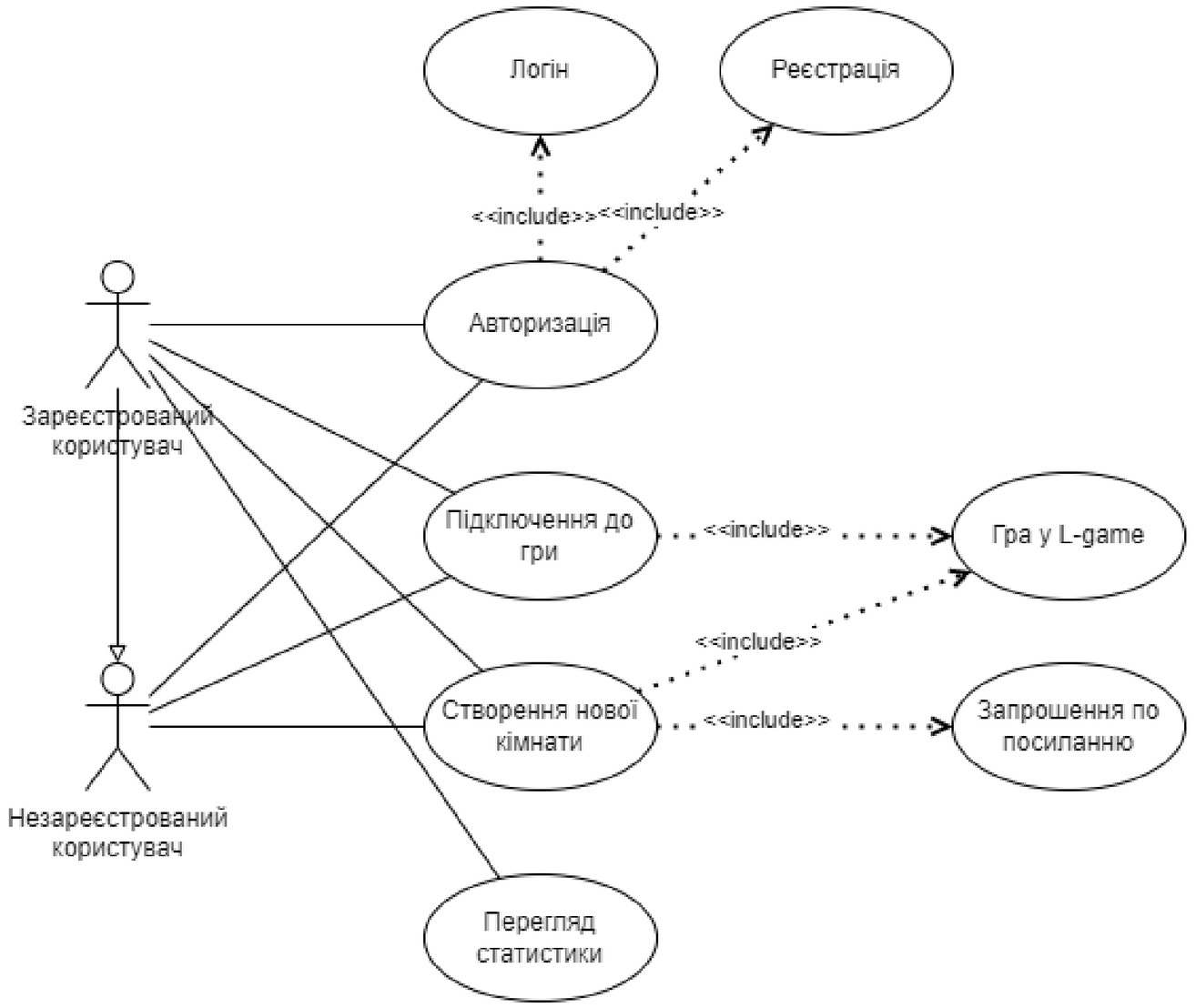


Рисунок 2 – Діаграма варіантів використання

					КвРІП3.200118.01.01.Е8		
						Літера	Маса	Масштаб
Зм.	Арк.	№докум.	Підпис	Дата	Веб-сервіс для онлайн-гри у головоломку «L-game» на базі JS та Node.js			
Розробив		Бадьора Я. М		22.05.23		Аркуш 2	Аркушів 3	
Керівник		Форкун Ю. В.		22.05.23				
Консульт.								
Н.Контр.		Праворська Н.І		22.05.23		ХНУ, ІПЗс-20-1		
Зав.каф.		Бедратюк Л.П.		22.05.23				

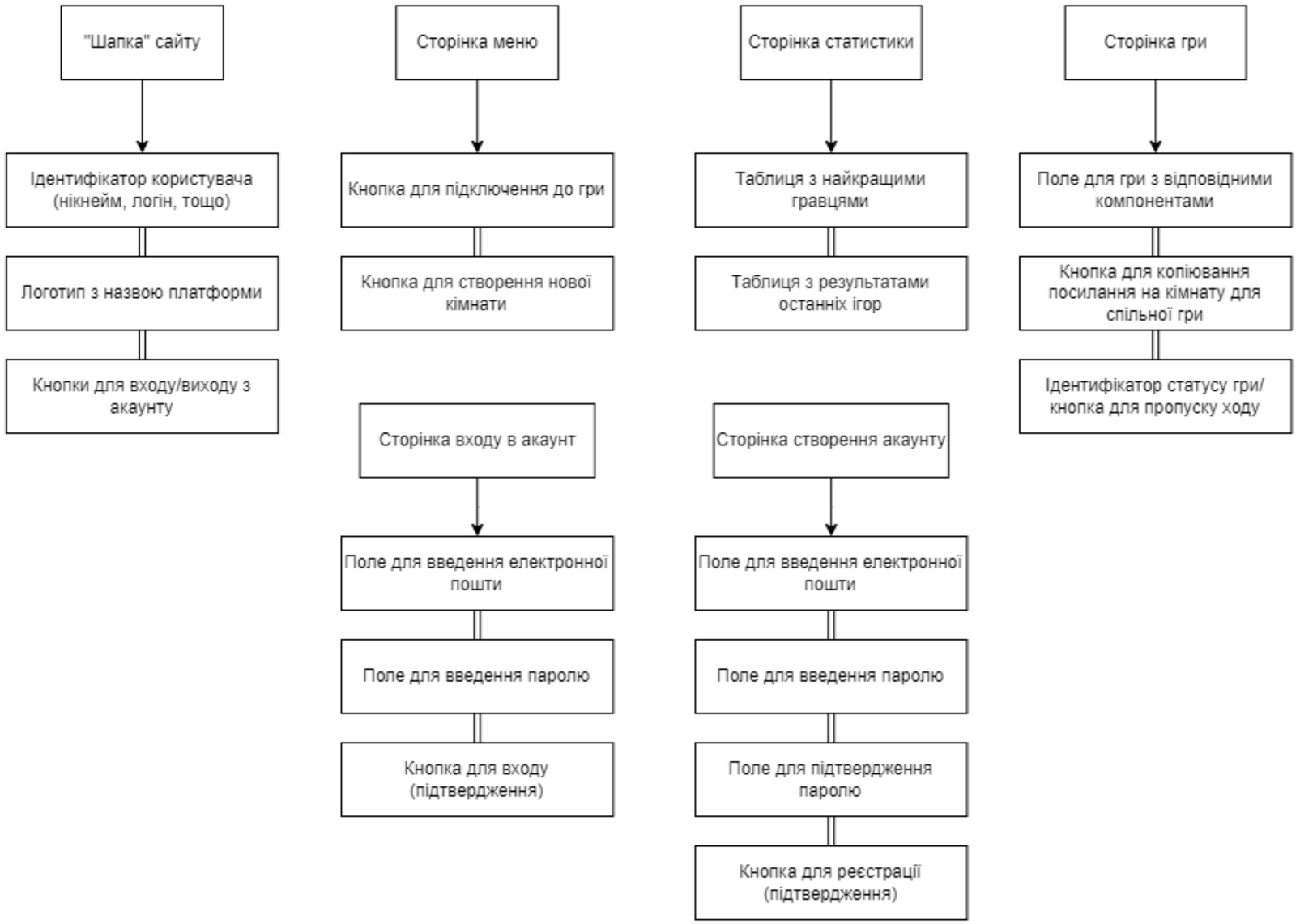


Рисунок 3 – Схема структури інтерфейсу користувача

					КвРІПЗ.200118.01.01.Е8		
Зм.	Арк.	№докум.	Підпис	Дата	Веб-сервіс для онлайн-гри у головоломку «L-game» на базі JS та Node.js	Літера	Маса / Масштаб
Розробив		Бадьора Я. М		22.05.23			
Керівник		Форкун Ю. В.		22.05.23		Аркуш 3	Аркушів 3
Консульт.							
Н.Контр.		Праворська Н.І		22.05.23		ХНУ, ІПЗс-20-1	
Зав.каф.		Бедратюк Л.П.		22.05.23			

Завідувачу кафедри
інженерії програмного забезпечення
проф. Бедратюку Л. П.
студента групи ІПЗс-20-1
Бабюри Ярослава М.
Прізвище, ініціали

ЗАЯВА

Прошу закріпити за мною тему кваліфікаційної роботи освітнього ступеня «бакалавр» за спеціальністю 121 «Інженерія програмного забезпечення»: Веб-сервіс для онлайн-ігри у головоломку „Ь-game" на базі JS та NodeJS

(керівник роботи – Ядуцук Юлій Вікторович)
Прізвище, ім'я, по батькові

01.03.2023
Дата

Підпис студента

Завідувачу кафедри інженерії програмного забезпечення проф. Бедратюку Л. П.

здобувача вищої освіти

Бадьори Я. М.

Прізвище, ініціали

факультет ІТ, 3 курс, група ІПЗс-20-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності в Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений та надаю свою згоду на обробку й збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та/або Anti-Plagiarism) і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

02.06.23

дата

підпис

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ДЕКЛАРАЦІЯ УЧАСНИКА ОСВІТНЬОГО ПРОЦЕСУ

щодо дотримання академічної доброчесності

Цією декларацією я, Бордюга Ярослав Михайлович
Прізвище, ім'я, по батькові

здобувач вищої освіти (шифр та назва спец-ті, курс, академічна група)/ науково-педагогічний працівник (назва кафедри)

назва факультету

підтверджую, що ознайомився (- лась) з Положенням про систему забезпечення академічної доброчесності у Хмельницькому національному університеті та Кодексом академічної доброчесності і **зобов'язуюсь** дотримуватися їх вимог під час освітнього процесу, проведення наукової діяльності, виконання організаційно-адміністративних функцій тощо.

Усвідомлюю, що у разі порушення мною принципів академічної доброчесності нестиму відповідальність перед академічною спільнотою ХНУ згідно з нормами, визначеними Положенням про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, законодавства України.

«02» червня 2023 р.

Підпис

Mon Jun 05 11:39:02 EEST 2023, Хіврич Володимир Русланович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 9%**

	Документ		Сумарний збіг по Базі Даних	
ID: 114735 Назва: БКР Веб-сервіс для онлайн-гри у головоломку «L-game» на базі JS та Node.js Додано в БД: 2023-06-05 Автора: Бадьора Я.М. Керівники: Форкун Ю.В. к.т.н. доц. Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	80219	652	1407 (2%)	13 (2%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK

Ім'я користувача:
Кафедра ІПЗ

Дата перевірки:
05.06.2023 12:58:56 EEST

Дата звіту:
05.06.2023 13:02:34 EEST

ID перевірки:
1015430227

Тип перевірки:
Doc vs Internet + Library

ID користувача:
100005589

Назва документа: **Звіт_з_КвР_Бадьора_Ярослав_ІПЗс_20_1 плагіат**

Кількість сторінок: **64** Кількість слів: **14777** Кількість символів: **116969** Розмір файлу: **441.76 KB** ID файлу: **1015091034**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

11.9%
Схожість

Найбільша схожість: **5.28%** з джерелом з Бібліотеки (ID файлу: **1015045648**)

1.66% Джерела з Інтернету	331	Сторінка 66
11.4% Джерела з Бібліотеки	113	Сторінка 67

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	1
Підозріле форматування	48 сторінок

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатами звіту/звітів подібності щодо роботи, продукованими програмно-технічним засобом (ами) перевірки текстів на плагіат:

Назва: «Веб-сервіс для онлайн-гри у головоломку «L-game» на базі JS та Node.js»

Автор: Бадьора Ярослав Михайлович
Спеціальність: 121 – Інженерія програмного забезпечення
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Науковий керівник: Форкун Юрій Вікторович, кандидат технічних наук, доцент
Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	**відповідає**
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої й електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того, як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) у тексті кваліфікаційної роботи системами перевірки на плагіат виявлено схожість з деякими документами в частині загальновживаних обов'язкових словосполучень у стандартних бланках (титулка, відомість документів), у структурі змісту, назвах розділів/підрозділів тощо, у назвах публікацій у переліку джерел посилання;

2) в якості запозичень системою було зафіксовано деякі послідовності вихідного коду і посиланання на бібліотеки, які є стандартними мовними конструкціями програмування та не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;

3) усі запозичення є фрагментарними або мають належним чином оформленні посилання;

4) виявлені модифікації тексту не впливають на відсоток схожості.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів ідентичності/ схожості, складає 1% та 11,9% і адресується до рамки записки в зв'язку з помилкою відображення форматування, що, з урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь кваліфікаційної роботи.

Дата _____

Завідувач кафедри _____ Леонід БЕДРАТЮК

Гарант освітньої програми _____ Леонід БЕДРАТЮК

Керівник кваліфікаційної роботи _____ Юрій ФОРКУН

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
освітнього ступеня «Бакалавр»

Дипломник Бадьора Ярослав Михайлович

Тема Веб-сервіс для онлайн-гри у головоломку «L-game» на базі JS та Node.js

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3; кількість сторінок записки 111

1.Короткий зміст пояснювальної записки та прийнятих рішень У процесі виконання кваліфікаційної роботи була проведено детальне аналітичне дослідження предметної області, включаючи вивчення всіх функціональних і нефункціональних вимог. Було ретельно проаналізовано існуючі програмні рішення на ринку, їх переваги та недоліки, що підтверджує актуальність розробки нового програмного забезпечення. В процесі вивчення були враховані різні інструменти для реалізації запропонованих рішень, що дозволило успішно розробити програмне забезпечення. Крім того, було проведено тестування розробленого програмного забезпечення, під час якого було підтверджено його коректну роботу та готовність до впровадження.

2. Висновок про відповідність роботи поставленому завданню Кваліфікаційна робота виконана відповідно до поставленого завдання та з дотриманням всіх вимог.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступній частині було показано, що обрана тема є актуальною, а також визначено мету та завдання дипломного проектування. Перший розділ присвячений детальному аналізу предметної області, в якому було розглянуто наявні рішення та встановлені функціональні та нефункціональні вимоги до розроблюваного програмного забезпечення. У другому розділі проведений аналіз сучасних архітектур, в якому були розглянуті переваги та недоліки, і визначено, що система буде базуватися на монолітній архітектурі та моделі клієнт-сервер. Третій розділ був присвячений підготовці всіх залежностей для програмного коду і практичній розробці програмних модулів з описом їх особливостей, що призвело до створення програмного продукту. У цьому розділі також проведено модульне тестування системи згідно з функціональними вимогами, що підтвердило правильну роботу програми.

4. Позитивні сторони роботи Застосунок має приємний та сучасний дизайс в стилі мінімалізму та в спокійних тонах. Елементи інтерфейсу працюють плавно та мають гарні анімовані дії.

5. Негативні сторони роботи Відсутня гра проти компютера.

6. Оцінка графічного оформлення та пояснювальної записки Графічне оформлення виконано відповідно до теми кваліфікаційної роботи та подано у вигляді діаграм і рисунків. Пояснювальна записка оформлена згідно вимог чинних стандартів.

7. Відгук про кваліфікаційну роботу в цілому Кваліфікаційна робота заслуговує позитивної оцінки, оскільки матеріал пояснювальної записки демонструє чітку структуру, логічну послідовність, зрозумілий та доступний виклад. Це дозволяє чітко сприйняти вміст роботи, який гармонійно вписується в контекст тематики проектування. Графічний матеріал, наданий у роботі, надає можливість візуально ознайомитись з деталями проектування системи.

8. Інші зауваження

9. Оцінка кваліфікаційної роботи Кваліфікаційна робота виконана у повному обсязі, відповідає поставленій задачі та заслуговує на оцінку «задовільно».

РЕЦЕНЗЕНТ: Кисіль Наталія Миколаївна
зав. кафедри КІЕС, к.ф.-м.н., доц.

"____" _____ 2023 р. _____
(підпис)