

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Веб-технології»)

на тему:

«Вебплатформа симулятора мініігор»

Виконав студент групи ВТ-22-1

Бігун Роман Васильович

Керівник роботи:

ЄФРЕМОВ Юрій Миколайович

Рецензент:

ВАКАЛЮК Тетяна Анатоліївна

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна Вакалюк

«27» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **БІГУН Роман Васильович**

Керівник роботи: **ЄФРЕМОВ Юрій Миколайович**

Тема роботи: **«Вебплатформа симулятора мініігор»**,

затверджена Наказом закладу вищої освіти від **«27» лютого 2026 р., №92с**

Вихідні дані для роботи:

- об'єкт дослідження: процес взаємодії користувача з ігровою вебплатформою;
- предмет дослідження: методи та засоби розробки клієнт-серверних вебзастосунків на основі JavaScript-технологій і мікросервісної архітектури.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	ЄФРЕМОВ Ю.М.	01.03.2026р.	01.03.2026р.
2	ЄФРЕМОВ Ю.М.	01.03.2026р.	01.03.2026р.
3	ЄФРЕМОВ Ю.М.	01.03.2026р.	01.03.2026р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з пояснювальної записки та програмного продукту — вебплатформи симулятора мініігор. Пояснювальна записка містить 65 сторінок основного тексту, 27 рисунків та 10 таблиць; список використаних джерел налічує 36 найменувань.

У роботі розглянуто розробку вебплатформи симулятора мініігор із казино-подібними механіками та віртуальною валютою. Мета роботи — створення сучасної інформаційної системи, що забезпечує інтерактивний ігровий процес, обробку результатів і керування балансом користувачів у єдиному середовищі.

Об'єктом дослідження є процес взаємодії користувача з ігровою вебплатформою, а предметом — методи та засоби розробки клієнт-серверних вебзастосунків із використанням сучасних JavaScript-технологій і мікросервісної архітектури.

Для реалізації обрано стек Vue з інструментом Vite для клієнтської частини та Node.js (Express) — для серверних мікросервісів; взаємодія відбувається через RESTful API, автентифікація — за допомогою JWT. Спроектовано базу даних і реалізовано серверне обчислення результатів, що гарантує чесність ігрового процесу.

Результатом є функціональний прототип вебплатформи зі стабільною роботою ігрових сценаріїв, централізованим керуванням балансом і безпечною взаємодією. Практична цінність — основа для подальшого створення повноцінного ігрового сервісу.

Ключові слова: вебплатформа, симулятор мініігор, Vue, Vite, Node.js, REST API, JWT, мікросервіси.

					ІПЗ.КР.Б – 121 – 26 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Бігун Р.В.			Вебплатформа симулятор мініігор Пояснювальна записка	Лім.	Арк.	Аркушів
Перевір.							3	86
Керівник		Єфремов Ю.М.				ФІКТ Гр. ВТ-22-1		
Н. контр.								
Зав. каф.								

ABSTRACT

The bachelor's graduation qualification work consists of an explanatory note and a software product - the web platform "Minigame Simulator". The explanatory note contains 65 pages of the main text, 27 figures and 10 tables; the list of used sources includes 36 items.

The work considers the development of the web platform "Minigame Simulator" with casino-like mechanics and virtual currency. The purpose of the work is to create a modern information system that provides an interactive game process, processing results and managing user balances in a single environment.

The object of the study is the process of user interaction with the game web platform, and the subject is methods and tools for developing client-server web applications using modern JavaScript technologies and microservice architecture.

The Vue stack with the Vite tool for the client part and Node.js (Express) for server microservices was chosen for implementation; interaction occurs via RESTful API, authentication — using JWT. A database has been designed and server-side calculation of results has been implemented, which guarantees the integrity of the gaming process.

The result is a functional prototype of a web platform with stable operation of game scenarios, centralized balance management and secure interaction. Practical value — the basis for further creation of a full-fledged gaming service.

Keywords: web platform, minigame simulator, Vue, Vite, Node.js, REST API, JWT, microservices.

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ док.м.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ ІГРОВИХ ВЕБПЛАТФОРМ	10
1.1 Постановка задачі	10
1.2 Аналіз аналогів програмного продукту.....	12
1.3 Архітектура вебплатформи та інструментальні засоби її реалізації	18
ВИСНОВКИ ДО ПЕРШОГО РОЗДІЛУ	22
РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБПЛАТФОРМИ СИМУЛЯТОРА МІНІГОР	24
2.1 Визначення варіантів використання системи	24
2.2 Розробка бази даних системи	30
2.3 Проєктування та реалізація алгоритмів роботи системи.....	33
2.4 Структура модулів та реалізація вебплатформи	38
ВИСНОВКИ ДО ДРУГОГО РОЗДІЛУ	42
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБПЛАТФОРМОЮ СИМУЛЯТОРА МІНІГОР.....	44
3.1 Порядок встановлення та налаштування системи.....	44
3.2 Структура інтерфейсу вебплатформи.....	47
3.3 Тестування роботи програмного комплексу	56
ВИСНОВКИ ДО ТРЕТЬОГО РОЗДІЛУ	59
ВИСНОВКИ.....	60
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	62
ДОДАТКИ	66

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

UI – User Interface (користувацький інтерфейс)

UI/UX – Проектування інтерфейсу та досвіду користувача

API – Application Programming Interface (програмний інтерфейс застосунків)

DB – Database (база даних)

SQL – Structured Query Language (структурована мова запитів)

JSON – JavaScript Object Notation (текстовий формат обміну даними)

REST – Representational State Transfer (архітектурний стиль API)

OAuth – Open Authorization (протокол авторизації)

JWT – JSON Web Token (веб-токен доступу)

PostgreSQL – Реляційна система керування базами даних

Vue – фреймворк для побудови вебінтерфейсів

TypeScript – Типізована надбудова над JavaScript

SPA – Single Page Application (односторінковий вебзастосунок)

SSG – Static Site Generation (статична генерація сторінок)

SSE – Server-Sent Events (односпрямовані події від сервера)

UUID – Universally Unique Identifier (універсальний унікальний ідентифікатор)

CRUD – Create, Read, Update, Delete (базові операції з даними)

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Сьогодні індустрія цифрових розваг стрімко розвивається та виходить на новий рівень інтерактивності. У сучасних умовах вебтехнології стали основним середовищем для створення та поширення ігрових сервісів. Для користувача онлайн-ігри — це вже не просто розвага, а можливість взаємодіяти з динамічними системами, отримувати миттєвий результат і керувати власними ресурсами у віртуальному середовищі. Тому створення власної ігрової вебплатформи є актуальним і перспективним напрямом розробки програмного забезпечення.

Розробка вебплатформи симулятора мініігор є сучасним рішенням, яке поєднує інтерактивні ігрові механіки та систему управління віртуальними ресурсами. Такий підхід дозволяє створити централізоване середовище для запуску ігрових сценаріїв, обробки результатів і контролю балансу користувачів. Це вже не просто вебсайт, а комплексна система, що забезпечує чесність і прозорість обчислень, а також безпечну взаємодію між клієнтом і сервером.

Для користувачів платформа надає зручний доступ до ігрового процесу: від запуску мініігор до отримання результатів і керування власним балансом у декілька дій. Розробники та адміністратори, у свою чергу, отримують інструменти для налаштування ігрових механік, контролю системи та аналізу активності користувачів. Автоматизація цих процесів дозволяє підвищити ефективність роботи системи та забезпечити стабільність її функціонування, передавши обчислювальні та організаційні завдання програмному забезпеченню.

Світовий ринок інтерактивних онлайн-розваг з ігровими та парі-механіками демонструє стійке зростання, а основним каналом доступу до таких сервісів став вебпереглядач, який не потребує встановлення додаткового програмного забезпечення й однаково працює на настільних і мобільних пристроях. Це висуває на перший план задачу побудови продуктивних, безпечних і масштабованих вебзастосунків реального часу, здатних

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

опрацьовувати ігрові події та операції з внутрішнім балансом без затримок і помилок.

Аналіз наявних рішень показує, що багатьом ігровим платформам притаманні спільні недоліки. Насамперед це непрозорість обчислення результатів, через яку користувач не може переконатися в чесності гри, та перевантаженість інтерфейсу рекламними блоками, що ускладнює доступ до основного функціоналу. Не менш важливою є проблема безпечного керування віртуальними ресурсами користувача, адже будь-яка можливість вплинути на результат гри чи на баланс із боку клієнта підриває довіру до сервісу. Саме ці невирішені питання й визначають актуальність роботи.

З погляду інженерії програмного забезпечення розробка такої платформи є показовою задачею, оскільки поєднує проєктування клієнт-серверної взаємодії, мікросервісної архітектури, реляційної бази даних, механізмів автентифікації та обчислень реального часу. Ключовим архітектурним рішенням є перенесення всієї ігрової логіки й операцій з балансом на сервер, де їх неможливо підмінити, що забезпечує чесність і безпеку системи. Тому дослідження методів побудови подібних застосунків має як практичне, так і навчально-методичне значення.

Метою кваліфікаційної роботи є створення вебплатформи симулятора мініігор — клієнт-серверної інформаційної системи, що поєднує набір інтерактивних мініігор з казино-подібними механіками, віртуальну ігрову валюту та централізоване серверне обчислення результатів, яке гарантує чесність ігрового процесу.

Для досягнення поставленої мети необхідно вирішити низку завдань. Передусім — проаналізувати предметну область і наявні на ринку ігрові вебплатформи, щоб визначити їхній типовий функціонал та недоліки, і на цій основі сформулювати функціональні й нефункціональні вимоги до власної системи. Далі належить обґрунтувати вибір клієнт-серверної архітектури й технологічного стеку, спроектувати структуру бази даних та алгоритми роботи мініігор. На етапі реалізації потрібно розробити клієнтську частину у вигляді

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

односторінкового вебзастосунку на Vue 3 з підтримкою двох мов інтерфейсу та серверну частину у вигляді мікросервісів на Node.js із захищеним REST API і подіями реального часу, а також реалізувати ігрову економіку — баланс, предмети, кейси, інвентар та аукціон між гравцями. Завершальними завданнями є тестування програмного комплексу й розгортання його у хмарному середовищі.

Об'єктом дослідження є процес взаємодії користувача з ігровою вебплатформою в режимі реального часу.

Предметом дослідження є методи та засоби розробки клієнт-серверних вебзастосунків із мікросервісною архітектурою на основі JavaScript-технологій.

Методами дослідження у роботі є методи аналізу та порівняння існуючих програмних рішень, системного аналізу вимог, об'єктно-орієнтованого проєктування, моделювання за допомогою UML [16], а також експериментальні методи тестування програмного забезпечення.

Практичне значення одержаних результатів полягає у створенні повнофункціонального прототипу ігрової вебплатформи, яка відтворює механіки азартних ігор без використання реальних коштів, що робить її безпечним розважальним сервісом і водночас основою для подальшого розвитку комерційного продукту. Реалізована архітектура може бути використана як приклад побудови мікросервісних систем із віртуальною економікою.

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ ІГРОВИХ ВЕБПЛАТФОРМ

1.1 Постановка задачі

Вебплатформа симулятор мініігор — це інтерактивна інформаційно-розважальна система, що об'єднує набір коротких ігор в єдину цифрову платформу з профілем користувача, прогресом та статистикою. Особливістю проєкту є використання механік, схожих на ігри з казино-сайтів, однак без реальних грошей: усі ставки та винагороди виконуються у віртуальній валюті, яка існує лише в межах системи. Такий підхід дозволяє зберегти динаміку та азарт ігрового процесу, водночас виключивши фінансові ризики та забезпечивши контрольовану ігрову економіку.

Вебплатформа симулятора мініігор складається з декількох програмних модулів, які забезпечують різноманітні функції та можливості.

Модуль каталогу мініігор (Frontend) забезпечує відображення списку ігор, їх правил, рівнів складності та доступних режимів. Реалізує єдиний сценарій взаємодії. Клієнтська частина розробляється на Vue + Vite з компонентним підходом та адаптивним інтерфейсом.

Модуль ігрових сесій та логіки (Game Session Service) відповідає за запуск/завершення сесії, фіксацію подій гри та підрахунок результату (очки, коефіцієнт, виграш/програш, час). Модуль реалізовано як мікросервіс на Node.js, який надає API для створення сесії, оновлення її стану та завершення з поверненням фінального результату.

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

Модуль віртуальної валюти та економіки (Wallet/Balance Service) реалізує систему “монет/кредитів”: стартовий баланс, правила ставок, ліміти, нарахування “виграшів” і списання “витрат” виключно у віртуальній валюті. Також забезпечує історію операцій у межах системи для прозорого відображення змін балансу та запобігання некоректним операціям.

Модуль статистики та результатів (Stats Service) забезпечує збереження історії спроб, найкращих результатів, середніх значень, серій перемог/поразок та агрегованої статистики як по окремих мінііграх, так і по всьому профілю. Мікросервіс на Node.js надає API для отримання статистики, формування підсумкових показників та підготовки даних для UI.

Модуль рейтингів і лідербордів (Leaderboard Service) формує таблиці лідерів за кожною грою та загальні рейтинги (наприклад за рекордами або за сумарним показником). Передбачає фільтрацію за періодом і режимом гри, а також механізми оновлення рейтингу після завершення кожної сесії.

Модуль користувачів та доступу (Auth/User Service) відповідає за реєстрацію/авторизацію, управління профілем, ролі доступу (користувач/адміністратор) та прив’язку балансу, статистики і результатів до конкретного акаунта. Передбачено захист API через токени та контроль доступу до операцій, пов’язаних із профілем і валютою.

Ключовим етапом розробки є проектування архітектури системи з розділенням на клієнтську частину та незалежні серверні компоненти. Frontend на Vue + Vite реалізує UI/UX, маршрутизацію, інтерактивні екрани мініігор та відображення статистики, а взаємодія з сервером виконується через HTTP API. Backend реалізовано як набір мікросервісів на Node.js, що дозволяє ізолювати домени (користувачі, сесії, баланс, статистика, рейтинги), спростити масштабування та підтримку, а також забезпечити надійність обчислень результатів і коректність операцій з віртуальною валютою.

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

На етапі бізнес-логіки реалізовується обробка ігрових подій, правила нарахування/списання віртуальної валюти, фіксація результатів та автоматичне оновлення статистики і лідербордів. Завершальним етапом є тестування API та інтеграції між мікросервісами, перевірка коректності розрахунків, стабільності збереження даних, адаптивності інтерфейсу та оптимізація продуктивності клієнтської частини.

У результаті виконання поставленого плану має бути створена повнофункціональна вебплатформа симулятор мініігор, яка забезпечує зручний доступ до мініігор для користувачів, безпечну (в межах системи) ігрову економіку без реальних грошей, збереження прогресу та статистики, а також надає адміністративні інструменти керування правилами та аналізом активності.

1.2 Аналіз аналогів програмного продукту

Розробка вебплатформи симулятора мініігор потребує аналізу існуючих рішень на ринку. У рамках дослідження було проведено порівняльний огляд українських ігрових платформ, які мають подібні механіки (швидкі раунди, бонусні системи, турніри, “колесо”, місії), щоб виділити сильні UX-практики та типові недоліки, важливі для проєктування власного продукту. За аналогії обрано реальні казино та одну гру, за жанром клікер, для реалізації безкоштовності свого аналогу, оскільки на їх основі побудовано симулятор. Для огляду відібрано чотири найпопулярніші та найрепрезентативніші платформи — GGBET, BETON, Gorilla і 777.ua, — які охоплюють різні підходи до організації ігрового процесу.

Першим аналогом розглянемо GGBET — українську платформу, що поєднує онлайн-казино та ставки на спорт із великим каталогом ігор (рис. 1.1)

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

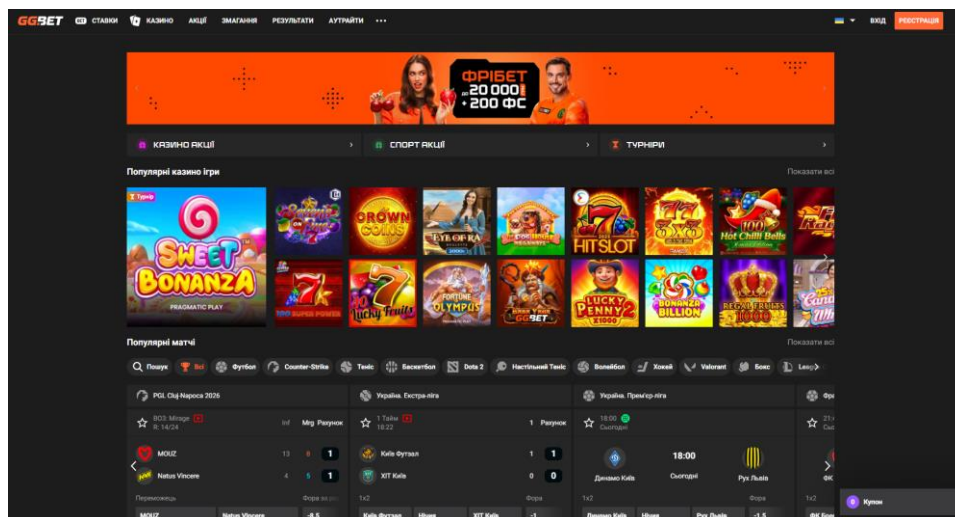


Рис. 1.1 – Вебсайт GGBET.ua

Серед переваг GGBET — поєднання казино та ставок на спорт в одному сервісі, великий асортимент ігор і категоризація слотів (популярні, джекпот, бонусні тощо), що спрощує первинний пошук контенту; декларована орієнтація на безпеку та «відповідальну гру» формує довіру користувача. Водночас сайт перевантажений інформаційними блоками й пропозиціями, через що у новачка зростає когнітивне навантаження — складніше швидко знайти потрібну гру чи режим без «шуму» від численних розділів. До того ж значна частина контенту подана як промо, а не як інструментальна аналітика для користувача — історія сесій, зрозумілі метрики прогресу та деталізована статистика по кожній мінігрі.

Другим аналогом є BETON — сервіс із насиченою гейміфікацією: турнірами, місіями, лутбоксами та «колесом фортуни» (рис. 1.2)

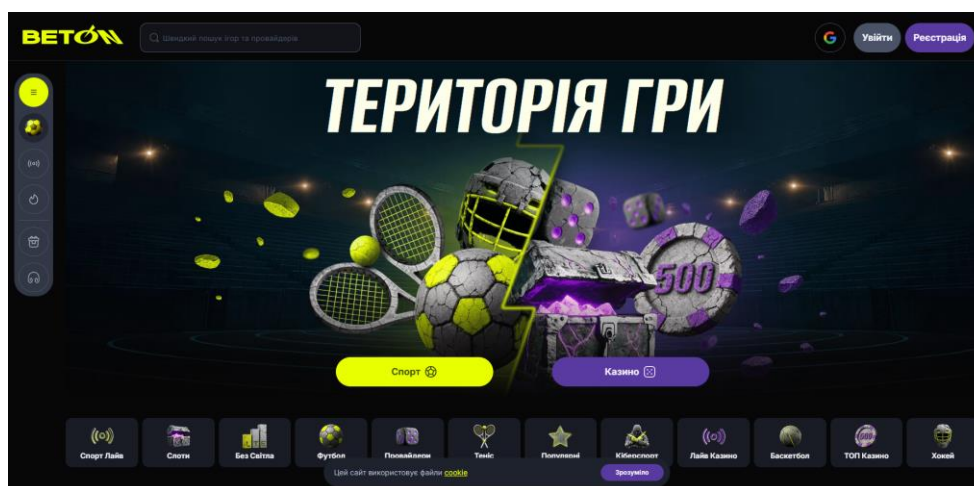


Рис. 1.2 – Вебсайт Beton.ua

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

BETON активно використовує гейміфікацію — турніри, місії, «магазин», лутбокси, колесо та рівні, — що підвищує залучення й час взаємодії з продуктом, а сам сервіс прагне бути «монолітним». Проте на сторінках домінують великі промо-блоки та заклики до реєстрації, що погіршує досвід користувача, який хоче спершу ознайомитися з продуктом без зайвих кроків. Видимість ключових елементів гри конкурує з бонусами й акціями, тому навігація стає менш чистою та більш маркетинговою, а велика кількість гейміфікаційних модулів без чіткої ієрархії ускладнює інтерфейс і дезорієнтує користувача.

Третій аналог, Gorilla, робить акцент на швидких іграх і простому, лаконічному вході в ігровий процес (рис. 1.3)

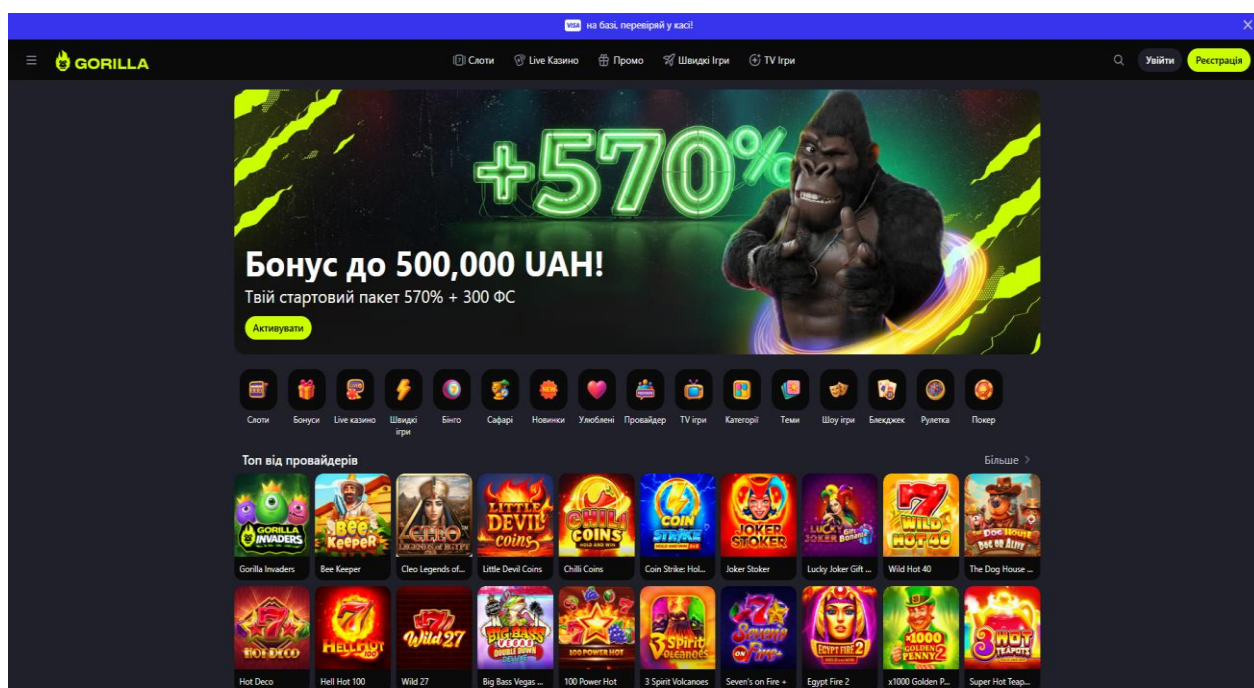


Рис. 1.3 – Вебсайт Gorilla.ua

Сильною стороною Gorilla є акцент на швидких іграх і простому, лаконічному вході в ігровий процес — короткий цикл взаємодії, зрозумілий інтерфейс і мінімум зайвих дій, що є цінним орієнтиром для симулятора мініігор. Натомість у відкритих оглядах згадується відсутність мобільного застосунку за наявності лише адаптивної версії, що знижує зручність для частини аудиторії; крім того, платформі бракує глибини персональної

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		14

аналітики — користувачу часто не вистачає історії сесій, зрозумілих метрик прогресу та інструментів порівняння результатів за окремими іграми.

Четвертим аналогом є 777.ua — відома платформа з широким каталогом ігор, турнірами та зручною мобільною версією (рис. 1.4)

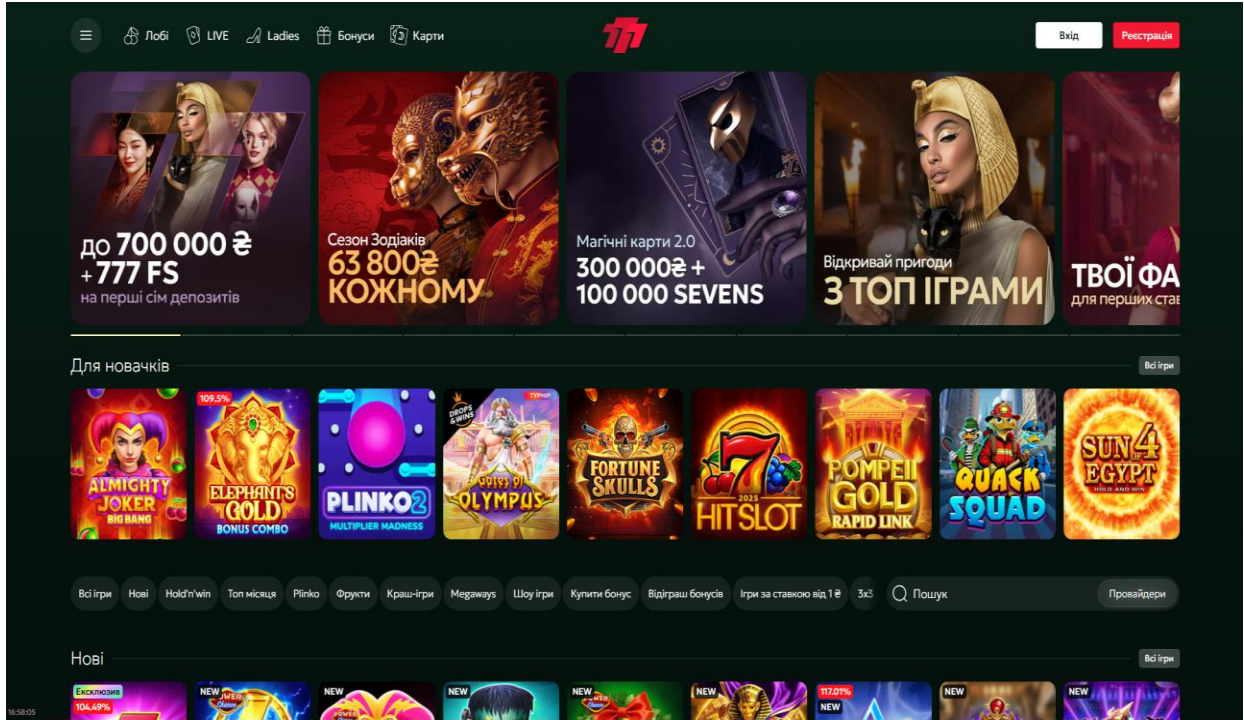


Рис. 1.4 – Вебсайт 777.ua

Перевагою 777.ua є наявність турнірів і розіграшів, широкий каталог ігор та мобільна орієнтація (адаптивність і коректна робота на смартфонах), що демонструє, як утримання користувача будується на івентах і регулярних активностях. Проте інформаційна подача перевантажена довгими описовими блоками, через що складніше швидко дійти до конкретної дії — обрати гру, переглянути статистику чи знайти правила; акцент тут робиться на каталозі й промо, а не на прозорому поясненні механік та зручних інструментах аналітики, як-от історія спроб, деталізація результатів і графіки прогресу.

Узагальнюючи аналіз існуючих рішень, складемо порівняльну таблицю їх функціональних можливостей та нашого проєкту (таблиця 1.1).

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

Таблиця 1.1.

Порівняння ігрових платформ-аналогів

Характеристика	GGBET. ua	Beton. ua	Gorilla. ua	777.u a	Розроблюва на веб платформа
Каталог мініігор та категоризація	+	+	+	+	+
Зрозумілий “швидкий старт” гри (мінімум кроків)	+-	+-	+	+-	+
Детальна статистика по кожній грі (історія спроб, метрики)	+-	+-	-	+-	+
Збереження історії сесій (раунди/спроби з результатами)	+-	+-	-	+-	+
Лідерборди / рейтинги	+-	+	-	+-	+
Пошук по іграм/контенту	+-	+-	-	+-	+
Оптимізація для мобільних пристроїв	+	+	+-	+	+
Модульність/розширюваність (додавання нових ігор без переробки системи)	+-	+-	-	+-	+

Провівши детальний аналіз існуючих рішень на ринку ігрових платформ з казино-подібними механіками (GGBET, BETON, Gorilla, 777.ua), можна зробити висновок, що більшість таких сервісів мають спільні проблеми у

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

побудові користувацького досвіду та подачі функціоналу. Платформи часто перевантажені промо-блоками, бонусними банерами та другорядними сценаріями, через що основна дія — швидко знайти потрібну гру, зрозуміти правила та отримати результат — ускладнюється для новачка. Крім того, у багатьох аналогів спостерігається недостатня глибина аналітики: статистика подається поверхнево, історія ігрових сесій не завжди доступна у зручному вигляді, а правила нарахування результату та “винагород” залишаються неочевидними, що знижує прозорість взаємодії з системою.

Розроблювана платформа має поєднати найкращі практики розглянутих сервісів, але при цьому усунути критичні помилки в UX/UI, забезпечивши створення цілісної екосистеми мініігор. Додатковою концептуальною відмінністю є те, що система працює без реальних коштів: механіки подібні до казино, однак ставки й винагороди реалізуються у віртуальній валюті, що дозволяє сфокусуватися на ігровому процесі, статистиці та чесних правилах без фінансових сценаріїв.

На відміну від існуючих рішень, орієнтованих на маркетингові й промо-сторінки, розроблюваний симулятор матиме ключові відмінності та переваги завдяки сучасній архітектурі: frontend на Vue + Vite забезпечить високу швидкість завантаження, компонентний підхід та адаптивний інтерфейс без зайвого візуального “шуму”, а backend у вигляді мікросервісів на Node.js гарантуватиме коректність бізнес-логіки та розрахунків. Мікросервісна структура дозволить ізолювати домени, що спростить масштабування й підтримку, а також мінімізує ризики помилок при зміні правил окремих мініігор. Окремо передбачено реалізацію детальної історії сесій, зрозумілої статистики по кожній грі, прозорих правил зміни балансу, пошуку/фільтрації по каталогу мініігор та швидкого оновлення рейтингів — саме тих компонентів, які в аналогах або реалізовані частково, або поступаються за зручністю.

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

1.3 Архітектура вебплатформи та інструментальні засоби її реалізації

Для реалізації вебплатформи симулятора мініігор обрано гібридну клієнт-серверну архітектуру з вебклієнтом та backend, реалізованим як набір мікросервісів на Node.js [14], що взаємодіють через RESTful API [7]. Такий вибір обумовлений потребою поєднати високу інтерактивність ігрового інтерфейсу (швидкі раунди, миттєве оновлення UI, короткі сесії) з надійною серверною бізнес-логікою, яка контролює розрахунок результатів, облік віртуального балансу та збереження статистики. Альтернативою була монолітна архітектура, проте її відкинуто: монолітне ядро ускладнює незалежне масштабування навантажених компонентів, підвищує ризик каскадних відмов і робить супровід громіздким у міру зростання кількості мініігор. Натомість система проєктується модульно, де кожен сервіс відповідає за окремий домен — користувачі, ігрові сесії, «гаманець», статистика та лідерборди, — що дає змогу розвивати, тестувати й розгортати ці частини окремо.

Клієнтську частину реалізовано на фреймворку Vue [36] з інструментом збірки Vite [31]. Vue обрано через компонентний підхід і реактивну модель даних, що особливо зручні для ігрових екранів, де стан інтерфейсу часто змінюється у відповідь на дії гравця та події сервера. Компонентна організація дає змогу уніфікувати екрани мініігор і повторно використовувати елементи для профілю, історії сесій, статистики й таблиць лідерів, скорочуючи дублювання коду. Vite забезпечує миттєвий запуск проєкту та гаряче перезавантаження (HMR) під час розробки завдяки нативним ES-модулям, а у продакшн-збірці — оптимізований бандл із розщепленням коду. Взаємодія з backend виконується через HTTP-запити до RESTful endpoint'ів за допомогою HTTP-клієнта на основі Fetch API [6], а дані відображаються у реальному часі в межах сесії користувача.

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

Backend реалізовано у вигляді мікросервісної архітектури на платформі Node.js [14] із вебфреймворком Express [5]. Вибір Node.js зумовлений єдиною мовою (JavaScript/TypeScript) для клієнта й сервера, що спрощує розробку та обмін кодом, а також неблокувальною подієвою моделлю введення-виведення, яка добре пасує до застосунків реального часу з численними короткими запитами. Основний сервіс Game Session Service керує життєвим циклом кожної ігрової спроби: створення сесії, фіксація параметрів, завершення та повернення результату. Сервіс Wallet/Balance Service відповідає за операції з віртуальною валютою — перевірку достатності балансу, списання ставки, нарахування виграшу та ведення історії транзакцій. Сервіс Stats Service агрегує дані й формує метрики, а Leaderboard Service підтримує рейтинги за кожною грою та загальні лідерборди з можливістю фільтрації. Такий розподіл відповідальності підвищує надійність системи, спрощує тестування, а також дозволяє масштабувати найнавантаженіші компоненти незалежно від інших.

Для забезпечення безпеки та контролю доступу використовується механізм автентифікації на базі JWT [26]. Після успішного входу користувач отримує токен, який надалі передається у заголовок авторизації при зверненні до захищених RESTful endpoint'ів. Це дозволяє реалізувати stateless-підхід без зберігання серверних сесій, спрощує горизонтальне масштабування сервісів і забезпечує контроль прав доступу на рівні ролей (користувач/адміністратор). Публічні ресурси (наприклад, каталог мініігор і загальні рейтинги) доступні без токена, тоді як приватні (баланс, історія сесій, персональна статистика) захищені JWT-авторизацією.

Оскільки система працює зі структурованими даними та взаємопов'язаними операціями (завершення гри одночасно фіксує результат, змінює баланс і впливає на статистику та рейтинги), ключовим архітектурним аспектом є узгодженість бізнес-логіки. Тому обчислення результату й зміни балансу виконуються виключно на сервері, а клієнт виступає лише інтерфейсом взаємодії, що мінімізує ризик підміни результатів. Додатково RESTful API дозволяє стандартизувати контракти між фронтом і сервісами

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

та забезпечити прогнозовану інтеграцію. Архітектурна модель передбачає чітке розділення системи на логічні зони доступу: публічна зона включає каталог мініігор, правила та перегляд загальних лідербордів; приватна зона надає доступ до профілю, балансу віртуальної валюти, історії сесій і персональної статистики; за потреби виділяється адміністративна зона, захищена ролями, яка дозволяє керувати переліком мініігор, параметрами правил (ліміти ставок, коефіцієнти) та переглядати агреговану аналітику активності. Таке розмежування забезпечує безпеку, масштабованість і зручність користування для різних категорій користувачів.

Обрана архітектура реалізується за допомогою сучасного клієнт-серверного технологічного стеку, що забезпечує продуктивність інтерфейсу, надійну серверну логіку та безпечну взаємодію компонентів системи. Клієнтську частину побудовано на фреймворку Vue.js з інструментом збірки Vite, що дає компонентний підхід, швидкий запуск проєкту та гаряче перезавантаження під час розробки; взаємодія із сервером здійснюється HTTP-клієнтом на основі Fetch API [6]. Серверну частину реалізовано на платформі Node.js [14] із вебфреймворком Express [5], а обмін даними між клієнтом і сервісами відбувається через RESTful API; захист приватних операцій забезпечує механізм JWT [26]. Як сховище даних застосовано реляційну СКБД PostgreSQL [21], а для уніфікації середовища розгортання — контейнеризацію за допомогою Docker [3] із оркестрацією через Docker Compose [2].

Вибір реляційної СКБД PostgreSQL [21] як основного сховища зумовлений тим, що платформа оперує взаємопов'язаними сутностями та грошовими значеннями, для яких критичні цілісність і транзакційність (підтримка властивостей ACID). Реляційна модель із зовнішніми ключами природно описує зв'язки «користувач — сесії — предмети — лоти», а транзакції гарантують узгодженість складених операцій (наприклад, купівлі на аукціоні), коли кошти й предмети мають змінитися одночасно або не змінитися взагалі. Для уніфікації середовища застосовано контейнеризацію Docker [3] з оркестрацією Docker Compose [2]: усе оточення — шлюз, база

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

даних і обидва сервіси — описано одним файлом, що дозволяє підняти ідентичну конфігурацію на будь-якій машині однією командою й усуває розбіжності між середовищами розробки та експлуатації.

Загалом обраний стек є внутрішньо узгодженим: єдина мова JavaScript/TypeScript на клієнті й сервері спрощує розробку та обмін типами даних, мікросервісний поділ забезпечує масштабованість і відмовостійкість, а RESTful API з JWT-автентифікацією дає простий, передбачуваний і безпечний контракт взаємодії. Перевагу REST над альтернативами (зокрема GraphQL) надано через простоту кешування, прозорість маршрутів і достатність для наявного набору операцій; для подій реального часу в іграх «гравець проти гравця» додатково застосовано односпрямований канал Server-Sent Events, що не потребує складнішого двостороннього протоколу.

Інструментальне забезпечення процесу розробки добиралося з огляду на продуктивність і зручність супроводу коду. Середовищами розробки слугували Visual Studio Code для клієнтської частини та WebStorm — для серверної; обидва редактори підтримують підсвічування синтаксису, рефакторинг і покрокове налагодження JavaScript/TypeScript. Контроль версій вівся за допомогою системи Git, що забезпечує історію змін, гілкування та можливість відкату. Перевірку коректності REST-запитів виконано інструментом Postman, який дозволяє формувати запити різних типів, передавати токени авторизації та аналізувати відповіді сервера. Якість коду підтримується статичним аналізатором ESLint [4] і форматувальником Prettier [22], а типобезпеку клієнтської частини гарантує TypeScript [28]. Для автономної розробки інтерфейсу без запущеного бекенду передбачено мок-режим на основі Mock Service Worker [11].

Вимоги до апаратного забезпечення є помірними. Для розробки й тестування достатньо двоядерного процесора рівня Intel Core i3 або AMD Ryzen 3, 8 ГБ оперативної пам'яті, 10–20 ГБ вільного місця на накопичувачі та операційної системи Windows 10/11, Linux чи macOS. Для комфортної паралельної роботи фронтенду й кількох мікросервісів у контейнерах

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

рекомендовано чотири- або шестиядерний процесор, 16 ГБ оперативної пам'яті та твердотільний накопичувач, що пришвидшує збірку проєкту й роботу бази даних. Кінцевому користувачеві потрібен лише персональний комп'ютер або смартфон із доступом до Інтернету, сучасний браузер (Chrome, Firefox або Edge) та з'єднання швидкістю від 5 Мбіт/с, оскільки вся ресурсомістка обробка виконується на серверній стороні, а клієнт відповідає лише за відображення інтерфейсу й анімацій. Такий розподіл навантаження робить платформу доступною для широкого кола користувачів незалежно від потужності їхніх пристроїв.

ВИСНОВКИ ДО ПЕРШОГО РОЗДІЛУ

У першому розділі сформульовано постановку задачі: визначено, що вебплатформа симулятор мініігор має відтворювати казино-подібні ігрові механіки на віртуальній валюті без використання реальних коштів, забезпечуючи інтерактивний ігровий процес, ведення балансу користувачів і безпечну взаємодію між клієнтом та сервером. На цій основі окреслено перелік функціональних можливостей майбутньої системи.

Проведений аналіз чотирьох наявних ігрових платформ виявив спільні недоліки — перевантаженість інтерфейсу промоматеріалами, непрозорість обчислення результатів і недостатню глибину персональної аналітики. Це дозволило визначити типовий функціонал сервісів такого класу та сформулювати вимоги до власної розробки, яка має усунути виявлені недоліки.

Обґрунтовано вибір клієнт-серверної архітектури з мікросервісним бекендом: винесення ігрової логіки та операцій з балансом на сервер забезпечує чесність гри й унеможливорює маніпуляції з боку клієнта, а поділ на незалежні сервіси спрощує масштабування й супровід системи.

Визначено технологічний стек реалізації — Vue з інструментом Vite для клієнтської частини, Node.js з Express для серверних мікросервісів, PostgreSQL як сховище даних і Docker для уніфікації середовища — та сформульовано

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

помірні вимоги до апаратного забезпечення для розробки й використання платформи.

Отримані в першому розділі результати — постановка задачі, висновки з аналізу аналогів, обґрунтована архітектура й технологічний стек — формують основу для проєктування системи у другому розділі.

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБПЛАТФОРМИ СИМУЛЯТОРА МІНІГОР

2.1 Визначення варіантів використання системи

На етапі проектування на основі аналізу предметної області, проведеного у першому розділі, було сформовано вимоги до програмного продукту та визначено сценарії взаємодії користувачів із системою. Вимоги поділено на функціональні (що система має робити) та нефункціональні (якими властивостями вона має володіти). Чітке формулювання вимог на цьому етапі є основою для подальшого проектування бази даних, серверного API та користувацького інтерфейсу, оскільки дозволяє узгодити очікувану поведінку системи ще до початку реалізації й знизити ризик проєктних помилок. Для наочного подання ролей користувачів і доступних їм функцій сценарії взаємодії формалізовано у вигляді діаграми варіантів використання за нотацією UML [16].

Функціональні вимоги визначають, що саме повинна виконувати система. Платформа має забезпечувати реєстрацію та автентифікацію користувача за нікнеймом і паролем, а також вхід через обліковий запис Google за протоколом OAuth 2.0 [25]. Каталог мініігор та їх правила доступні для перегляду без обов’язкової авторизації, тоді як ставки у віртуальній валюті К приймаються лише від авторизованих гравців і супроводжуються миттєвим серверним розрахунком результату. Ігрова частина охоплює п’ять азартних мініігор — Dice, Roulette, Mines, Plinko та Coin Flip (битви між гравцями в реальному часі), — а також відкриття кейсів із випадковими предметами шести рівнів рідкості, що накопичуються в інвентарі. Додатково передбачено аукціон для торгівлі предметами між гравцями з комісією платформи та гру-клікер з окремою валютою, системою покращень і конвертацією накопичених коїнів в основну валюту.

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

Кожен користувач має доступ до особистого профілю з балансом, агрегованою статистикою за кожною грою та системою досягнень і може миттєво поповнити баланс через платіжну систему Google Pay. Для всіх відвідувачів відображається спільна стрічка останніх дропів і великих виграшів, а інтерфейс підтримує дві мови — українську та англійську. Адміністратор отримує окрему панель керування, яка дозволяє переглядати й шукати користувачів, коригувати баланси, блокувати та розблоковувати акаунти, переглядати історію поповнень і журнал власних дій, видавати досягнення та аналізувати статистику ігор і випадінь предметів.

Нефункціональні вимоги описують якісні характеристики системи. Клієнтську частину реалізовано як односторінковий застосунок із миттєвою навігацією між сторінками без перезавантаження та адаптивним інтерфейсом для настільних і мобільних пристроїв. Усі результати ігор обчислюються виключно на сервері, тож клієнт не може вплинути на результат або підмінити його, а операції зі зміни балансу виконуються атомарно й залишаються коректними за конкурентних запитів. Серверну частину спроектовано з можливістю незалежного масштабування окремих сервісів, доступ до приватних операцій захищено токенами JWT [26], а паролі зберігаються лише у вигляді криптографічних хешів.

У системі визначено трьох акторів. Гість — неавторизований відвідувач, який може переглядати каталог ігор, правила, стрічку дропів та загальнодоступні сторінки, а також зареєструватися чи увійти. Користувач — авторизований гравець, якому доступні ставки в мініігри, кейси, інвентар, аукціон, клікер, профіль зі статистикою. Адміністратор — користувач із роллю `admin`, який додатково керує користувачами, балансами, досягненнями та аналітикою платформи.

Діаграму варіантів використання наведено на рис. 2.1.

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		25



Рис. 2.1 – Діаграма варіантів використання системи

Опис основних прецедентів наведено в табл. 2.1 – 2.7.

Першим і базовим прецедентом є автентифікація, від якої залежить доступ до всіх приватних функцій платформи. У таблиці 2.1 наведено опис цього варіанту використання: основних акторів та сценарій створення облікового запису і входу, зокрема через обліковий запис Google. Цей

прецедент визначає, як гість стає авторизованим користувачем і отримує JWT для подальших захищених запитів.

Таблиця 2.1.

Опис прецеденту «Автентифікація»

Назва варіанту використання	Автентифікація
Основні актори	Гість
Короткий опис	Гість створює обліковий запис (нікнейм + пароль; пароль хешується алгоритмом bcrypt) або входить у наявний. Альтернативний потік — вхід через обліковий запис Google: сервер перевіряє ID-токен Google та створює/знаходить відповідного користувача. У відповідь видається JWT строком дії 7 діб, який клієнт додає до всіх захищених запитів.

Центральним сценарієм платформи є участь користувача в мінігрі зі ставкою у віртуальній валюті. У таблиці 2.2 описано цей прецедент — вибір гри, задання параметрів раунду та серверне списання ставки з нарахуванням виграшу. Саме він задає типовий цикл взаємодії гравця з ігровою логікою, спільний для більшості мініігор.

Таблиця 2.2.

Опис прецеденту «Гра в мінігрі зі ставкою»

Назва варіанту використання	Гра в мінігрі зі ставкою
Основні актори	Користувач
Короткий опис	Користувач обирає гру (Dice, Roulette, Mines, Plinko), задає розмір ставки та параметри раунду (поріг Roll Over, кількість мін, рівень ризику, ставки на полі рулетки). Сервер атомарно списує ставку, генерує випадковий результат, нараховує виграш і оновлює статистику гравця.

Окремий тип ігрової взаємодії — гра «гравець проти гравця», реалізована у вигляді битви Coin Flip. У таблиці 2.3 подано опис цього прецеденту: створення або приєднання до лобі, підтвердження готовності, відлік і визначення переможця з виплатою подвоєної ставки. На відміну від

решти ігор, тут результат стосується одразу двох користувачів, а оновлення стану надходять у реальному часі.

Таблиця 2.3.

Опис прецеденту «Битва Coin Flip»

Назва варіанту використання	Битва Coin Flip
Основні актори	Користувач
Короткий опис	Користувач створює лобі зі ставкою та стороною монети (або «випадковою») чи приєднується до відкритого лобі. Після підтвердження готовності обома гравцями запускається 3-секундний відлік, сервер розіграє сторону монети та виплачує переможцю подвоєну ставку. Оновлення стану битви надходять клієнтам у реальному часі через SSE.

Економіка предметів платформи починається з відкриття кейсів. У таблиці 2.4 описано відповідний прецедент — купівлю кейса, зважений випадковий вибір предмета за таблицею ймовірностей та додавання здобутого предмета до інвентаря. Цей сценарій формує наповнення інвентаря, що згодом використовується в аукціоні та клікері.

Таблиця 2.4.

Опис прецеденту «Відкриття кейсу»

Назва варіанту використання	Відкриття кейсу
Основні актори	Користувач
Короткий опис	Користувач купує кейс (Starter, Premium або Elite). Сервер списує вартість, виконує зважений випадковий вибір предмета за таблицею ймовірностей кейсу, додає предмет до інвентаря та публікує подію у стрічку дропів.

Здобуті предмети користувач може продавати іншим гравцям через внутрішній аукціон. У таблиці 2.5 наведено опис цього прецеденту: виставлення лота, купівлю іншим гравцем у межах однієї транзакції з утриманням комісії платформи та можливість зняти власний лот. Прецедент демонструє узгоджене переміщення коштів і предметів між акаунтами без ризику втрати даних.

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

Таблиця 2.5.

Опис прецеденту «Торгівля на аукціоні»

Назва варіанту використання	Торгівля на аукціоні
Основні актори	Користувач
Короткий опис	Продавець виставляє предмет зі свого інвентаря за довільною ціною. Покупець купує лот: у межах однієї транзакції БД кошти списуються з покупця, продавцю нараховується 95 % ціни (5 % — комісія платформи), предмет переходить до покупця, лот закривається. Продавець може зняти власний лот із продажу.

Альтернативний спосіб накопичення ресурсів реалізує гра «Клікер» з окремою валютою — коїнами. У таблиці 2.6 описано цей прецедент: накопичення коїнів кліками й автоматичним доходом, придбання покращень і конвертацію коїнів в основну валюту. Завдяки конвертації клікер інтегрується із загальною ігровою економікою платформи.

Таблиця 2.6.

Опис прецеденту «Гра «Клікер» і конвертація валюти»

Назва варіанту використання	Гра «Клікер» і конвертація валюти
Основні актори	Користувач
Короткий опис	Користувач накопичує коїни кліками та автоматичним доходом, купує покращення за коїни або предмети з інвентаря, після чого конвертує коїни в основну валюту К партіями по 1000 коїнів за 1 К.

Завершує перелік прецедентів адміністрування платформи, доступне лише користувачам із роллю адміністратора. У таблиці 2.7 подано опис цього варіанту використання: перегляд користувачів і деталей профілю, коригування балансів, видачу досягнень та перегляд історії дій. Цей прецедент забезпечує керованість і підзвітність системи.

Таблиця 2.7.

Опис прецеденту «Адміністрування платформи»

Назва варіанту використання	Адміністрування платформи
Основні актори	Адміністратор
Короткий опис	Адміністратор переглядає список користувачів та деталі профілю, коригує баланси, видає ручні досягнення, переглядає історію битв Coin Flip.

Сформовані варіанти використання повністю покривають функціональні вимоги та слугують основою для проєктування бази даних, серверного API та користувацького інтерфейсу в наступних підрозділах.

2.2 Розробка бази даних системи

Як систему збереження даних обрано реляційну СКБД PostgreSQL 15 [21]. Вибір обумовлений тим, що платформа оперує взаємопов'язаними сутностями та грошовими значеннями, для яких критичні цілісність і транзакційність (ACID). Грошові поля зберігаються у типі NUMERIC(20,2), що виключає похибки округлення, властиві числам із рухомою комою; первинні ключі більшості таблиць — UUID (gen_random_uuid()), що спрощує горизонтальне масштабування; для напівструктурованих даних (позиції мін, відкриті клітинки, рівні покращень) використано тип JSONB.

Таблиці логічно поділені між двома мікросервісами у межах спільної бази даних casino_db: до домену користувачів (сервіс auth-service) належать таблиці облікових записів, поповнень, досягнень і журналу дій адміністратора, а до ігрового домену (сервіс games-service) — таблиці сесій, битв, предметів, лотів аукціону, стану клікера та статистики. Схема створюється ідемпотентно під час старту кожного сервісу (CREATE TABLE IF NOT EXISTS із подальшими ALTER-міграціями), тому система не потребує окремого інструмента міграцій. Структуру бази даних наведено на рис. 2.2.

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		30

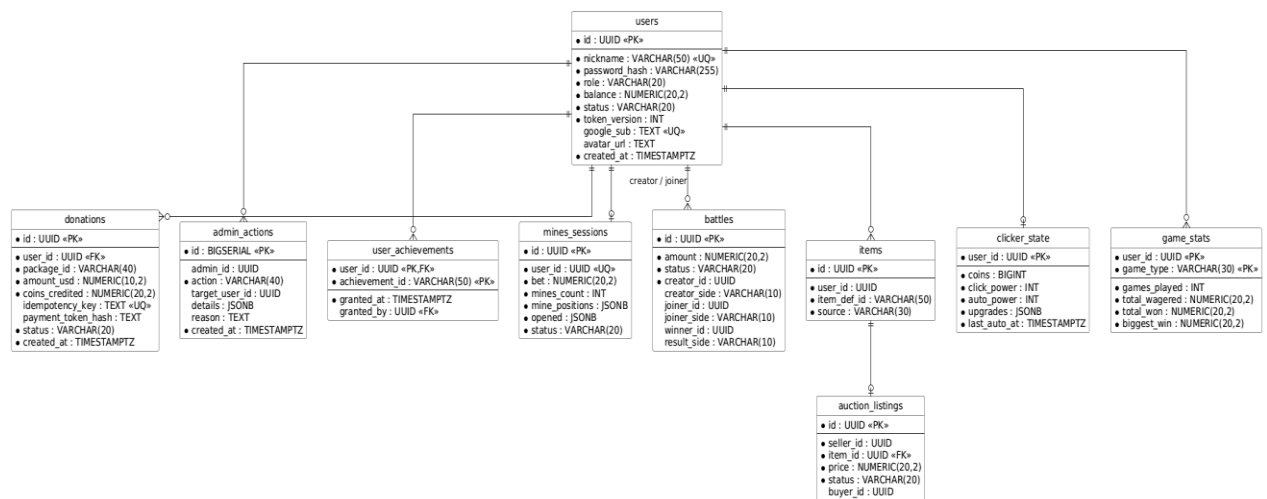


Рис. 2.2 – ER-діаграма бази даних системи

Центральною сутністю системи є таблиця `users`, що зберігає облікові записи та баланс. Кожен запис має унікальний ідентифікатор у форматі `UUID`, нікнейм і хеш пароля, обчислений алгоритмом `bcrypt` [23] (для облікових записів `Google` він порожній, оскільки автентифікацію виконує зовнішній провайдер). Тут же зберігаються роль доступу, поточний баланс віртуальної валюти у типі з фіксованою комою зі стартовим значенням 10 000 одиниць, ідентифікатор облікового запису `Google`, посилання на аватар і дата реєстрації. Для потреб модерації передбачено поля стану акаунта та версії токена, що дозволяють миттєво заблокувати користувача й анулювати всі його активні сесії.

Поповнення балансу фіксуються в таблиці `donations`, де для кожного платежу зберігаються користувач, ідентифікатор обраного пакета валюти, сума у доларах, обсяг нарахованої ігрової валюти, стан і дата операції. Окремий унікальний ключ ідемпотентності гарантує, що один платіж не буде нараховано двічі, а замість самого платіжного токена зберігається лише його хеш — для можливості аудиту без ризику розкриття платіжних даних.

Вручну видані адміністратором нагороди обліковуються в таблиці `user_achievements`, яка містить складений ключ «користувач — досягнення» разом із датою та автором видачі; натомість автоматичні досягнення в базі не зберігаються, а обчислюються динамічно за поточною статистикою гравця.

Усі дії адміністратора протоколюються в таблиці `admin_actions`, де фіксується, хто, коли та над яким користувачем виконав операцію, разом із деталями та причиною — це забезпечує підзвітність керування платформою.

Ігровий стан розподілено між кількома таблицями. Активні та завершені сесії гри Mines зберігають ставку, кількість мін і, що принципово важливо, координати мін у форматі JSONB, які зберігаються виключно на сервері й ніколи не передаються клієнту до завершення раунду. Битви Coin Flip описуються життєвим циклом зі станами очікування суперника, повного лобі, відліку та завершення, даними обох гравців, обраними сторонами монети й результатом розіграшу. Предмети гравців зберігаються в таблиці інвентаря, виставлені на продаж лоти — у таблиці аукціону, стан гри «Клікер» — в окремій таблиці, а агрегована статистика ставок за кожною грою — у таблиці `game_stats`, яка оновлюється атомарним запитом UPSERT.

Довідник предметів, що налічує 17 позицій шести рівнів рідкості — від звичайних вартістю 100 одиниць до міфічних вартістю 5 000 000 одиниць, — а також конфігурації кейсів зберігаються не в базі даних, а як константи коду ігрового сервісу. Такий підхід усуває потребу в довідникових таблицях, спрощує версіонування разом із кодом і гарантує узгодженість логіки випадіння та відображення предметів. Зведений перелік таблиць наведено в табл. 2.8.

Таблиця 2.8.

Перелік таблиць бази даних

Таблиця	Домен (сервіс)	Призначення
users	auth-service	Облікові записи, ролі, баланс віртуальної валюти
user_achievements	auth-service	Вручну видані досягнення
donations	auth-service	Історія поповнень балансу через Google Pay
mines_sessions	games-service	Сесії гри Mines із прихованими позиціями мін
battles	games-service	Битви Coin Flip та їх життєвий цикл
items	games-service	Інвентар предметів гравців
auction_listings	games-service	Лоти аукціону між гравцями
clicker_state	games-service	Стан гри «Клікер» (коіни, покращення)
game_stats	games-service	Агрегована статистика ставок за іграми

Між сутностями встановлено такі зв'язки: користувач може мати багато записів про поповнення, досягнення, предмети, ігрові сесії, статистику за іграми та битви, тоді як стан клікера й активна сесія Mines існують для нього в єдиному екземплярі; кожен предмет може бути пов'язаний щонайбільше з одним активним лотом аукціону, а дії над користувачами відображаються в журналі адміністратора. Така структура нормалізована, не містить дублювання даних і повністю покриває функціональні вимоги системи.

2.3 Проєктування та реалізація алгоритмів роботи системи

Базовий принцип системи — повне серверне обчислення результатів: клієнтська частина лише надсилає параметри раунду та анімує отриманий результат. Це унеможлиблює маніпуляції результатом на боці клієнта.

Усі ігрові маршрути реалізують спільний цикл обробки ставки. Клієнт надсилає POST-запит із параметрами раунду та токеном JWT у заголовку авторизації, після чого проміжний обробник перевіряє підпис токена й завантажує актуальний стан облікового запису. Далі функція списання балансу виконує атомарну операцію одним умовним запитом UPDATE, який зменшує баланс лише за умови його достатності; якщо коштів бракує, запит відхиляється без блокувань і станів гонки. Сервер генерує випадковий результат раунду, у разі виграшу нараховує виплату, а окрема функція оновлює агреговану статистику гравця. Наприкінці клієнт отримує відповідь і відтворює анімацію результату; якщо множник виплати сягає двадцяти і більше, подія публікується у спільну стрічку як великий виграш, що поділяється на рівні Big, Mega та Super Win — від 20×, 50× і 100× відповідно.

Послідовність взаємодії компонентів під час ставки показано на рис. 2.3.

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

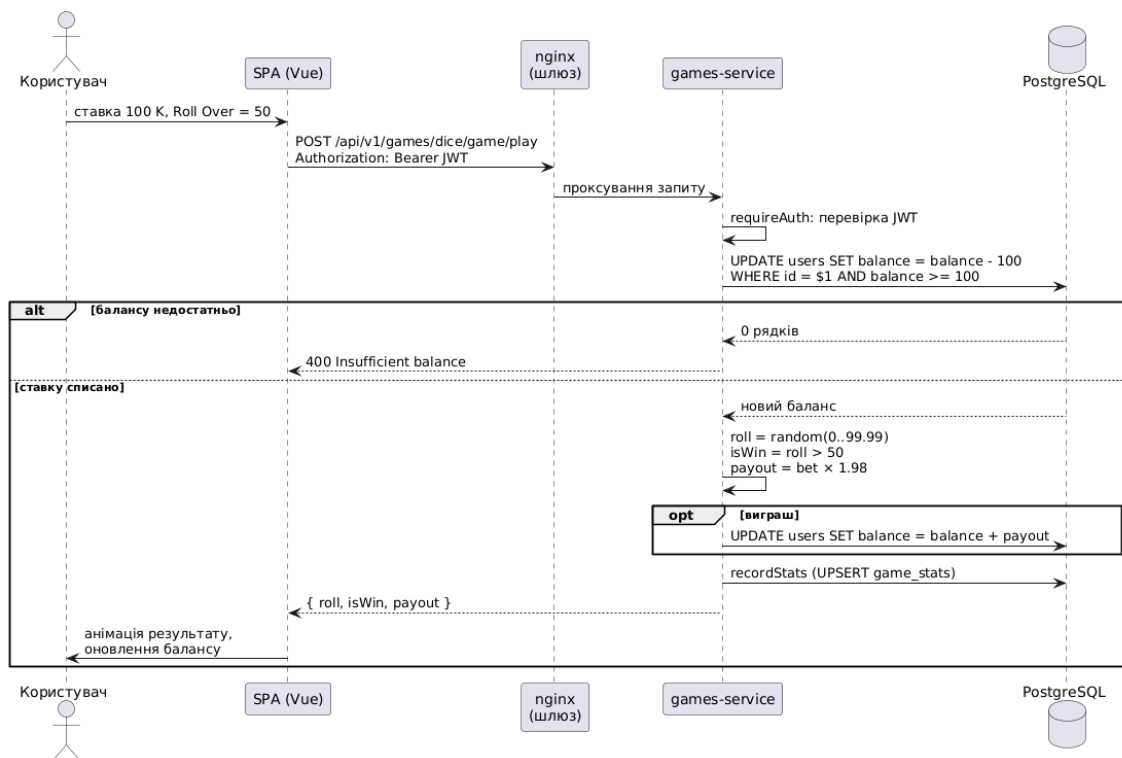


Рис. 2.3 – Діаграма послідовності обробки ставки у грі Dice

У грі Dice гравець задає поріг Roll Over R у діапазоні від 1 до 98. Сервер генерує випадкове число $roll$ від 0,00 до 99,99; раунд виграно, якщо $roll > R$. Імовірність виграшу та множник виплати обчислюються за формулами (2.1) і (2.2):

$$p = (100 - R) / 100 \quad (2.1)$$

$$k = (1 - h) / p \quad (2.2)$$

де R — обраний гравцем поріг; p — імовірність виграшу; $h = 0,01$ — комісія платформи (house edge 1 %); k — множник виплати. Наприклад, за $R = 50$ імовірність виграшу становить 0,495, а множник — 1,98. Чим вищий поріг, тим менша ймовірність і тим більший множник, отже математичне сподівання виплати для будь-якого порога однакове — 99 % ставки.

Гра Roulette реалізує європейську рулетку з 37 секторами (0–36). За один спін гравець може зробити довільну кількість ставок різних типів: на конкретне число (виплата $36\times$), на колір, парність чи половину поля ($2\times$), на дюжину або колонку ($3\times$). Сервер валідує кожен тип ставки за білим списком

ключів, списує сумарну ставку однією операцією, генерує номер сектора та повертає сумарний виграш за всіма ставками.

У грі Mines ігрове поле — 25 клітинок (5×5), гравець обирає кількість мін m від 1 до 24. Позиції мін генеруються перемішуванням Фішера–Єйтса [10] і зберігаються виключно на сервері у полі `mine_positions`, тому клієнт не може їх «підглянути». Кожна відкрита безпечна клітинка збільшує множник виплати, який обчислюється за формулою (2.3):

$$M(n) = 0,97 \cdot \prod_{i=0 \dots n-1} (25 - i) / (25 - m - i) \quad (2.3)$$

де n — кількість відкритих клітинок; m — кількість мін; добуток у знаменнику — ймовірність успішно відкрити n клітинок поспіль; коефіцієнт 0,97 задає рівень повернення гравцеві (RTP 97 %). Гравець може забрати виграш у будь-який момент (cash out); влучання в міну завершує раунд із втратою ставки та розкриттям усього поля.

У грі Plinko кількість рядів (8, 12 або 16) і рівень ризику (LOW / MEDIUM / HIGH) визначають таблицю множників комірок — від 0,2× до 1000×. Падіння кульки моделюється послідовністю незалежних рівноймовірних відхилень ліворуч або праворуч (по одному на ряд); індекс підсумкової комірки дорівнює кількості відхилень праворуч, що формує біноміальний розподіл — крайні комірки з великими множниками випадають рідко. За один запит можна запустити до 100 кульок: сервер повертає масив трас (бітових масок), за якими клієнт відтворює анімацію падіння кожної кульки.

Битви Coin Flip — єдина гра «гравець проти гравця». Життєвий цикл битви описується станами OPEN (очікування суперника), FULL (обидва гравці в лобі), COUNTDOWN (3-секундний відлік), FINISHED (розіграно), а також CANCELLED і ABANDONED для скасованих лобі. Ставка списується одразу під час створення або приєднання. Коли обидва гравці підтвердили готовність, сервер переводить битву в стан COUNTDOWN, а через 3 секунди генерує випадкову сторону монети `resultSide`. Переможцем стає гравець, чия сторона збіглася з результатом; якщо творець не обирав сторону (випадковий режим),

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

переможець обирається рівноймовірно. Переможцю нараховується подвоєна ставка. Перехід у стан FINISHED виконується умовним UPDATE ... WHERE status = 'COUNTDOWN', що гарантує одноразову виплату навіть за конкурентних запитів. Діаграму станів наведено на рис. 2.4.

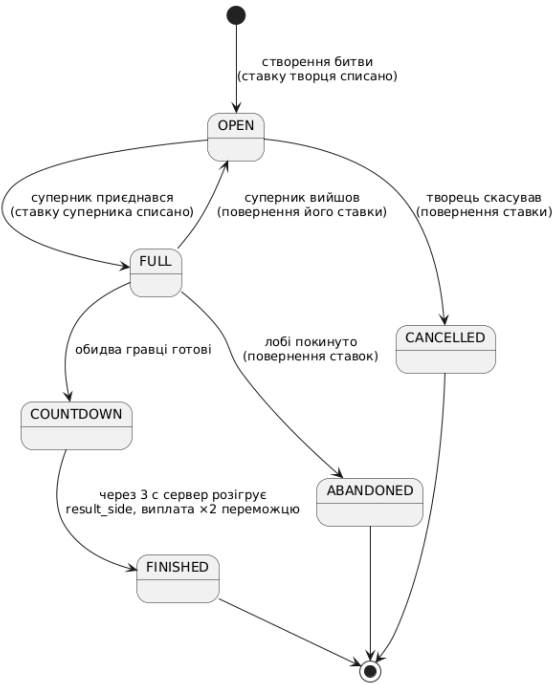


Рис. 2.4 – Діаграма станів битви Coin Flip

Оновлення стану битви доставляються клієнтам у реальному часі через Server-Sent Events [27]: клієнт відкриває EventSource на ендпоінт битви, а сервер надсилає подію після кожної зміни (приєднання, готовність, відлік, результат). На рівні шлюзу nginx для цього маршруту вимкнено буферування відповідей та збільшено таймаут читання, що є обов’язковою умовою коректної роботи SSE через зворотний проксі.

У грі з відкриттям кейсів, кожен кейс описується масивом пар «предмет — імовірність» (сума ймовірностей дорівнює одиниці). Розіграш виконується методом зваженого випадкового вибору: генерується число $r \in [0;1)$, після чого ймовірності додаються кумулятивно до першого перевищення r . Платформа містить три кейси — Starter (10 000 K), Premium (50 000 K) та Elite (100 000 K); що дорожчий кейс, то вищі шанси на предмети високої рідкості. Наприклад, у кейсі Elite предмет Cosmos Gem вартістю 5 000 000 K випадає з імовірністю

0,01 %. Отриманий предмет створюється в таблиці items, а подія публікується у стрічку дропів.

Гра «Клікер» використовує окрему валюту — коїни. Стан зберігається у clicker_state; автоматичний дохід нараховується «ліниво»: під час кожного звернення сервер обчислює інтервал від last_auto_at і додає auto_power × Δt коїнів, тож прогрес накопичується навіть офлайн. Покращення поділяються на два типи: за коїни та за предмети з інвентаря (предмети списуються під час купівлі) — усього п'ять рівнів від простих бустів до легендарних. Накопичені коїни конвертуються в основну валюту партіями по 1000 за формулою (2.4):

$$K = \lfloor C / 1000 \rfloor \quad (2.4)$$

де C — кількість коїнів; K — нараховані одиниці основної валюти; залишок коїнів зберігається.

На аукціоні виставлення лота перевіряє, що предмет належить продавцеві та ще не виставлений. Купівля виконується в межах однієї транзакції бази даних (BEGIN ... COMMIT): вибірка активного лота; умовне списання ціни з покупця (з перевіркою достатності балансу); нарахування продавцю 95 % ціни (5 % — комісія платформи); передача предмета покупцеві; закриття лота зі статусом SOLD. Будь-яка помилка на будь-якому кроці призводить до ROLLBACK, тому неможлива ситуація, коли кошти списано, а предмет не передано — навіть за одночасних спроб купівлі одного лота кількома гравцями.

У профілі є поле статистики та досягнень, після кожного раунду recordStats оновлює агрегати пари (користувач, гра). Ендпоінт /api/v1/stats/me паралельно агрегує статистику, баланс, інвентар і стан клікера та обчислює автоматичні досягнення за правилами: Millionaire — виграш від 1 000 000 K за раунд, Big Winner — від 100 000 K, Collector — від 10 предметів, Veteran — від 100 ігор, High Roller — сумарні ставки від 1 000 000 K, Mythic Hunter — наявність mythic-предмета тощо. Ручні досягнення (Special Award, Early Adopter, Loyal Player) видає адміністратор — лише вони зберігаються в таблиці user_achievements.

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

Під час поповнення балансу через Google Pay користувач обирає один із наперед визначених пакетів валюти й підтверджує оплату через Google Pay. Ціна та кількість валюти задаються виключно на сервері — клієнт надсилає лише ідентифікатор пакета, тож не може попросити довільну суму. Кожне поповнення супроводжується унікальним ключем ідемпотентності, який унеможлиблює повторне нарахування при повторному запиті, а платіжний токен зберігається тільки у вигляді хешу. Саме нарахування виконується атомарно в межах транзакції бази даних. Інтеграцію реалізовано в тестовому середовищі Google Pay; для прийому реальних платежів достатньо підключити платіжний провайдер (наприклад Stripe) без зміни серверної логіки.

Також реалізована стрічка подій реального часу, останні 50 дропів і великих виграшів зберігаються в кільцевому буфері в пам'яті games-service; клієнти періодично запитують ендпоінт `/api/v1/drops/recent` і відображають стрічку на головній сторінці. Такий підхід не навантажує базу даних подіями, що мають суто презентаційне значення.

2.4 Структура модулів та реалізація вебплатформи

Систему реалізовано за клієнт-серверною моделлю з мікросервісним бекендом [8, 1]. Запити клієнта проходять через єдину точку входу — шлюз nginx [12], який маршрутизує їх до відповідного сервісу: облікові операції — до auth-service (порт 5001), ігрові — до games-service (порт 5002). Обидва сервіси працюють зі спільною базою PostgreSQL. Для локального розгортання все середовище описано одним файлом `docker-compose.yml` [2, 3]: контейнери nginx, postgres:15-alpine та два сервіси. Схему взаємодії компонентів наведено на рис. 2.5.

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

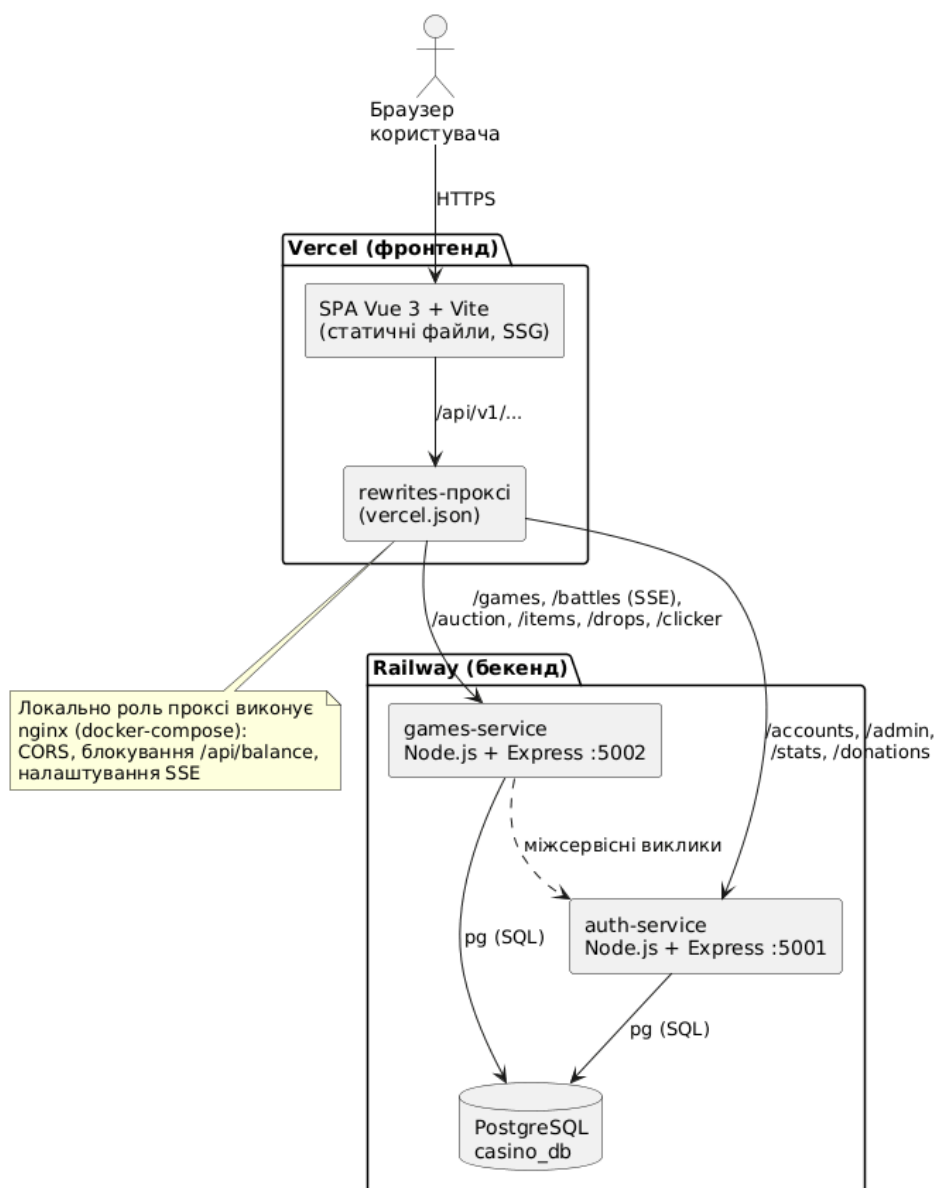


Рис. 2.5 – Схема взаємодії компонентів системи

Серверну частину реалізовано як набір мікросервісів — застосунок Node.js [14] на фреймворку Express [5] у форматі ES-модулів. Структура серверного коду:

dypлом_work_backend/	
├─ docker-compose.yml	— опис середовища розгортання
├─ nginx/nginx.conf	— конфігурація API-шлюзу
├─ auth-service/src/	
│ └─ index.js	— точка входу, реєстрація
маршрутів	
│ └─ db.js	— пул з'єднань, DDL-
ініціалізація схеми	
│ └─ middleware/auth.js	— requireAuth, requireAdmin
│ └─ routes/auth.js	— реєстрація, вхід, Google
OAuth, профіль	

```

|   └─ routes/admin.js      – адміністративні операції
|   └─ routes/stats.js     – агрегована статистика і
досягнення
└─ constants/achievements.js – довідник досягнень
└─ games-service/src/
   └─ index.js, db.js      – точка входу та схема ігрових
таблиць
   └─ middleware/auth.js   – JWT + операції з балансом
   └─ itemDefs.js          – довідник предметів (17
позицій)
└─ routes/                 – dice, roulette, mines,
plinko,
    battles, cases, clicker, auction, items, drops

```

Проміжний обробник `requireAuth` перевіряє заголовок `Authorization` та підпис `JWT` [26]; `requireAdmin` додатково звіряє роль користувача з базою даних (а не з токена), завдяки чому відкликання прав діє миттєво, не чекаючи завершення строку дії токена. Ключова функція грошових операцій — атомарне списання балансу:

```

export async function deductBalance(userId, amount) {
  const { rows } = await pool.query(
    `UPDATE users SET balance = balance - $1
     WHERE id = $2 AND balance >= $1 RETURNING balance`,
    [amount, userId]
  );
  if (!rows[0]) throw new Error('Insufficient balance');
  return parseFloat(rows[0].balance);
}

```

Умова `balance >= $1` всередині самого `UPDATE` робить перевірку та списання єдиною неподільною операцією: за конкурентних запитів СКБД сама серіалізує зміни рядка, і баланс не може стати від'ємним. Типовий ігровий маршрут на прикладі `Dice`:

```

router.post('/game/play', requireAuth, async (req, res) => {
  const bet = Number(req.body.bet);
  const rollOver = Number(req.body.rollOver);
  // ... валідація параметрів ...
  try { await deductBalance(req.user.id, bet); }
  catch { return res.status(400).json({ error: 'Insufficient
balance' }); }

  const roll = parseFloat((Math.random() *
99.99).toFixed(2));

```

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

```

const isWin = roll > rollOver;
const { payout: multiplier } = calcDicePayout(rollOver);
const payout = isWin ? parseFloat((bet *
multiplier).toFixed(2)) : 0;

if (isWin) await addBalance(req.user.id, payout);
recordStats(req.user.id, 'dice', bet, payout);
res.json({ roll, isWin, payout });
});

```

Шлюз nginx, окрім маршрутизації, виконує захисні функції: додає CORS-заголовки, повертає 403 на зовнішні звернення до внутрішнього ендпоінта /api/balance (він призначений лише для міжсервісної взаємодії) та забезпечує специфічні налаштування для SSE-маршруту битв (вимкнене буферування, подовжений таймаут).

Клієнтська частина — односторінковий застосунок на Vue 3 [36] із TypeScript [28], зібраний інструментом Vite [31]. Структура клієнтського коду:

```

dyplom_work_front/src/
├─ views/           – 16 сторінок: Home, Dice, Roulette,
Mines, Plinko,
├─                  CoinFlipLobby, CoinFlipBattle, Cases,
CaseDetail,
├─                  Clicker, Inventory, Auction, Profile,
Admin,
├─                  Login, Register
├─ stores/          – Pinia: auth, game, ui, bigwin
├─ router/          – маршрути та guards (requiresAuth,
requiresAdmin)
├─ core/http/       – apiClient: обгортка над Fetch, токен,
помилки
├─ api/             – типізовані виклики (games, battles)
├─ locales/         – переклади ua.ts / en.ts (vue-i18n)
├─ components/      – спільні компоненти (макети ігор,
модалки, тости)
├─ composables/     – повторно використовувана логіка
├─ utils/           – формат чисел, помилки, звук, SSE-клієнт
├─ styles/          – дизайн-токени та глобальні стилі

```

Стан застосунку керується сховищами Pinia [19]: сховище auth зберігає JWT і профіль користувача та інкапсулює оновлення балансу; ui — сповіщення (тости); bigwin — повноекранну анімацію великого виграшу.

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

Маршрутизатор Vue Router [35] захищає приватні сторінки навігаційними guard-ами:

```
r.beforeEach((to) => {
  const auth = useAuthStore()
  if (to.meta.requiresAuth && !auth.isAuthenticated) {
    return { name: 'login', query: { next: to.fullPath } }
  }
  if (to.meta.requiresAdmin && !auth.isAdmin) return { name:
'home' }
  return true
})
```

HTTP-клієнт реалізовано як тонку обгортку над Fetch API [6]: він автоматично додає заголовок Authorization, нормалізує помилки сервера до єдиного формату та опрацьовує відповіді 401 (розлогінення за простроченого токена). Інтернаціоналізація побудована на vue-i18n [34] із повними словниками української та англійської мов. Збірка виконується у режимі статичної генерації vite-ssg [32]: усі публічні сторінки рендеряться у HTML на етапі збірки, що покращує SEO-показники та швидкість першого відкриття; метатеги сторінок керуються бібліотекою @unhead/vue [29]. Для автономної розробки інтерфейсу передбачено мок-режим на основі Mock Service Worker [11], який перехоплює HTTP-запити та емулює відповіді бекенду.

Підсумкові механізми безпеки системи: хешування паролів bcrypt із фактором складності 10 [23]; JWT зі строком дії 7 діб [26]; перевірка ролі адміністратора з бази даних; повністю серверне обчислення результатів ігор; атомарні умовні UPDATE для всіх грошових операцій і переходів станів; транзакції для складених операцій аукціону; блокування внутрішніх ендпоінтів на рівні шлюзу. Перелічені заходи відповідають рекомендаціям OWASP щодо автентифікації та контролю доступу [18, 17].

ВИСНОВКИ ДО ДРУГОГО РОЗДІЛУ

У другому розділі виконано проектування системи. Визначено трьох акторів — гостя, користувача й адміністратора — та описано основні варіанти

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

використання: автентифікацію (зокрема через обліковий запис Google), ставки в мініігри, битви Coin Flip, відкриття кейсів, торгівлю на аукціоні, гру «Клікер» з конвертацією валюти, поповнення балансу та адміністрування платформи.

Спроектовано базу даних на основі PostgreSQL, таблиці якої логічно розподілено між двома мікросервісами. Для грошових значень застосовано точний числовий тип, для напівструктурованих даних — формат JSONB, а ідемпотентна ініціалізація схеми позбавила систему потреби в окремому інструменті міграцій.

Формалізовано та реалізовано алгоритми всіх ігрових механік — формули множників Dice і Mines, біноміальну модель Plinko, ставкову систему рулетки, життєвий цикл битв Coin Flip із доставкою подій у реальному часі, зважений випадковий дроп кейсів і транзакційну купівлю на аукціоні. Особливу увагу приділено коректності грошових операцій: атомарні умовні запити та транзакції унеможливають від’ємний баланс і подвійні виплати.

Описано структуру модулів клієнтської та серверної частин: єдину точку входу через шлюз, поділ серверного коду на сервіси обліку й ігор, організацію клієнта за сторінками, сховищами стану та спільними компонентами. Така структура забезпечує чітке розділення відповідальності й дозволяє додавати нові мініігри без зміни наявних компонентів.

Отримані проєктні рішення є безпосередньою основою для опису реалізованого інтерфейсу та тестування системи у третьому розділі.

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБПЛАТФОРМОЮ СИМУЛЯТОРА МІНІГОР

3.1 Порядок встановлення та налаштування системи

Для Розгортання вебплатформи передбачає підготовку середовища розробника, запуск серверної частини в контейнерах, налаштування клієнта та, за потреби, публікацію застосунку у хмарних сервісах. Для розгортання системи в середовищі розробки необхідні: Node.js версії 22 або новішої [14], менеджер пакетів npm, Docker Desktop або Docker Engine з підтримкою Compose [3, 2] та система контролю версій Git. Node.js потрібен для запуску клієнтських скриптів збірки та локального запуску сервісів поза контейнерами; Docker забезпечує ідентичність середовища на різних машинах і позбавляє потреби вручну встановлювати PostgreSQL; Git використовується для отримання вихідного коду репозиторіїв і подальшого оновлення. Перед першим запуском варто переконатися, що порти 80, 5001, 5002 та 5432 вільні, оскільки саме їх задіює середовище.

Серверна частина розгортається однією командою з кореня репозиторію бекенду:

```
git clone <репозиторій бекенду> && cd diplom_work_backend  
docker compose up --build
```

Compose-файл автоматично збирає образи й запускає чотири контейнери:

- nginx (порт 80) — API-шлюз, єдина точка входу для клієнта, що маршрутизує запити до відповідного сервісу та виконує захисні функції;
- postgres:15-alpine (порт 5432) — реляційна база даних casino_db;
- auth-service (порт 5001) — сервіс облікових записів, автентифікації та статистики;
- games-service (порт 5002) — ігровий сервіс, що обробляє ставки, кейси, аукціон, битви та клікер.

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

Під час першого запуску сервіси самостійно створюють схему бази даних (ідемпотентними запитами `CREATE TABLE IF NOT EXISTS` із подальшими `ALTER`-міграціями), тому окремий інструмент міграцій не потрібен, а повторний запуск не пошкоджує наявні дані. Тоді ж створюється службовий обліковий запис адміністратора (нікнейм `Admin`, пароль `admin`), пароль якого в робочому середовищі необхідно одразу змінити з міркувань безпеки. Успішність старту зручно перевірити командою `docker compose ps` (усі контейнери у стані «`running`» або «`healthy`»); у разі помилки журнал конкретного сервіса доступний командою `docker compose logs <ім'я сервіса>`, що допомагає швидко локалізувати проблему — наприклад, зайнятий порт чи невірний рядок підключення до бази даних.

Поведінка сервісів налаштовується змінними середовища, які задаються у файлі оточення або безпосередньо в `Compose`-файлі:

- `DATABASE_URL` — рядок підключення до PostgreSQL (хост, порт, ім'я бази, облікові дані);
- `JWT_SECRET` — секрет підпису токенів, спільний для обох сервісів; має бути однаковим, інакше токен, виданий одним сервісом, не пройде перевірку в іншому;
- `GOOGLE_CLIENT_ID` — ідентифікатор OAuth-клієнта Google (необов'язковий, потрібен лише для входу через Google);
- `PORT` — порт, на якому слухає сервіс;
- `AUTH_SERVICE_URL` — адреса `auth-service` для міжсервісних викликів `games-service`.

Клієнтська частина запускається в режимі розробки командами:

```
cd diplom_work_front
npm install
npm run dev    # http://localhost:5173
```

Команда `npm install` встановлює залежності проєкту, після чого `npm run dev` піднімає `dev`-сервер Vite із гарячим перезавантаженням за адресою `http://localhost:5173`. Для з'єднання з локальним бекендом у файлі `.env.local`

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

задається змінна `VITE_API_BASE_URL=http://localhost`, яка вказує клієнту базову адресу API-шлюзу. Передбачено також повністю автономний мок-режим (`VITE MOCK_API=1`): запити перехоплює Mock Service Worker [11], що дозволяє розробляти та демонструвати інтерфейс без запущеного бекенду й бази даних. Перед комітом або збіркою якість коду перевіряється командою `npm run check`, яка послідовно виконує перевірку типів (`vue-tsc`), лінтинг (`ESLint` [4], режим нульової толерантності до попереджень) і контрольну збірку; будь-яка помилка на цьому етапі зупиняє процес і не дає потрапити некоректному коду в репозиторій.

Для зручності розробки в репозиторії передбачено набір `npm`-скриптів: `npm run dev` запускає сервер розробки, `npm run build` виконує продакшн-збірку, `npm run preview` локально віддає зібраний застосунок, `npm test` запускає модульні тести, а `npm run check` об'єднує перевірку типів, лінтинг і збірку в одну команду. Такий набір скриптів стандартизує дії розробника й полегшує інтеграцію з конвеєрами безперервної інтеграції.

У разі проблем під час запуску корисні кілька типових перевірок. Якщо контейнер бази даних не стартує, найчастіша причина — зайнятий порт 5432 іншим екземпляром PostgreSQL; його потрібно зупинити або змінити проброшений порт у `Compose`-файлі. Якщо клієнт не бачить API, слід перевірити значення `VITE_API_BASE_URL` і доступність шлюзу `nginx` на порту 80. Помилки автентифікації між сервісами зазвичай свідчать про різні значення `JWT_SECRET`, а збій входу через Google — про відсутній або невірний `GOOGLE_CLIENT_ID`. Журнали кожного контейнера (`docker compose logs`) і перевірка стану (`docker compose ps`) дозволяють швидко встановити причину й усунути її.

Продакшн-розгортання виконано у хмарних сервісах, що позбавляє потреби в утриманні власного сервера. Бекенд розміщено на платформі Railway [24]: обидва сервіси збираються з власних `Dockerfile`, а базою даних слугує керований екземпляр PostgreSQL, який платформа надає та резервує автоматично. Фронтенд розміщено на Vercel [30]: команда `npm run build`

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

генерує карту сайту та виконує статичну генерацію публічних сторінок (vite-ssg [32]) у каталог dist, звідки Vercel роздає їх через глобальну мережу доставки вмісту. Звернення клієнта до API проксуються правилами rewrites у файлі vercel.json: запити /api/v1/accounts, /api/v1/admin, /api/v1/stats спрямовуються на auth-service, а /api/v1/games, /api/v1/battles, /api/v1/auction тощо — на games-service у Railway. Завдяки цьому клієнт працює з API в межах власного домену, що знімає проблеми CORS і спрощує налаштування безпеки. Після розгортання коректність роботи перевіряється відкриттям головної сторінки, реєстрацією тестового користувача та виконанням пробної ставки — якщо баланс змінюється, а результат відображається, систему розгорнуто правильно. Перелік запущених контейнерів серверної частини наведено на рис. 3.1.

```
postgres-1 | 2026-06-22 09:43:30.089 UTC [1] LOG:  listening on Unix socket "/var/run/postgres-1"
postgres-1 | 2026-06-22 09:43:30.096 UTC [57] LOG:  database system was shut down at 2026-06-22 09:43:30
postgres-1 | 2026-06-22 09:43:30.106 UTC [1] LOG:  database system is ready to accept connections
games-service-1 | Database initialized
games-service-1 | Games Service running on port 5002
auth-service-1 | ✓ Default admin created: nickname=Admin (password from ADMIN_PASSWORD)
auth-service-1 | Database initialized
auth-service-1 | Auth Service running on port 5001
postgres-1 | 2026-06-22 09:48:30.182 UTC [55] LOG:  checkpoint starting: time
```

Рис. 3.1 – Запущені контейнери серверної частини

3.2 Структура інтерфейсу вебплатформи

Інтерфейс платформи побудований за єдиною схемою: ліворуч — бічна панель навігації з переліком усіх ігор та розділів, угорі — поточний баланс користувача, перемикач мови та кнопки входу або профілю, у центральній частині — вміст активної сторінки. Інтерфейс виконано у темній стилістиці з акцентними кольорами рідкостей предметів; усі тексти доступні українською та англійською мовами.

Головна сторінка (рис. 3.2) відображає каталог мініігор у вигляді карток із назвою та коротким описом, а також живу стрічку останніх дропів і великих виграшів усіх гравців, яка оновлюється автоматично.

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		47

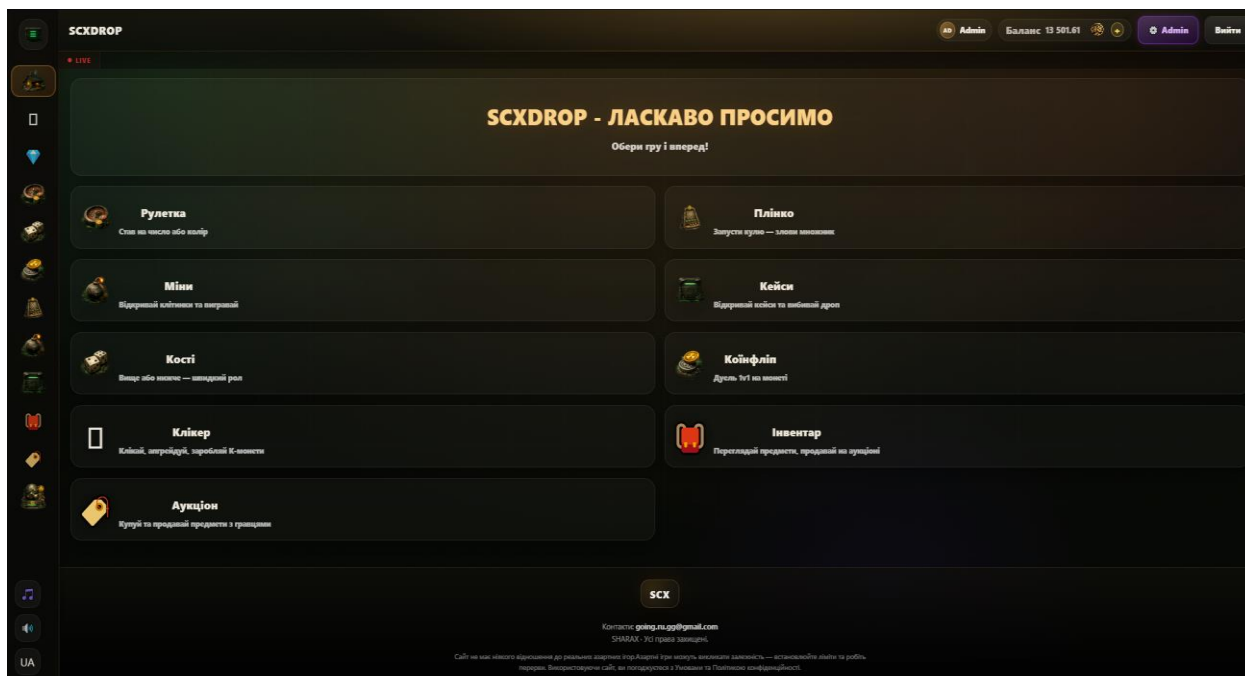


Рис. 3.2 – Головна сторінка вебплатформи

Сторінки реєстрації та входу (рис. 3.3, 3.4) містять форми з клієнтською валідацією полів та кнопку входу через Google. Після успішної автентифікації клієнт зберігає JWT і повертає користувача на попередню сторінку.

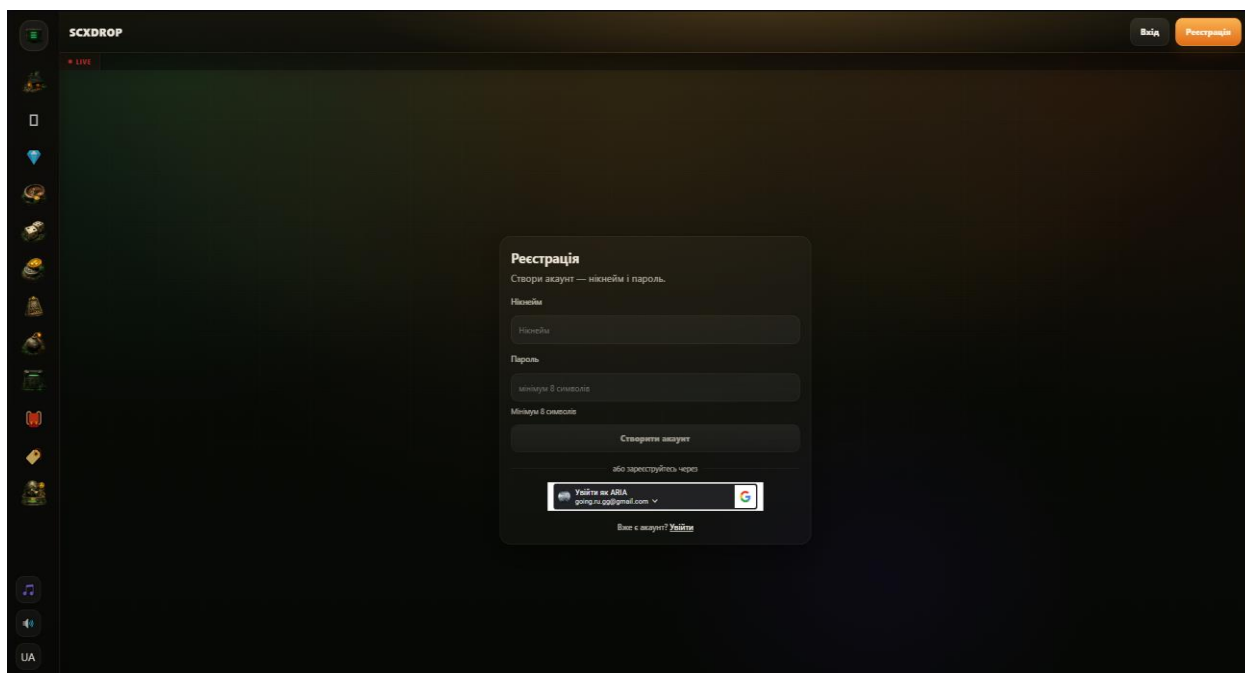


Рис. 3.3 – Сторінка реєстрації користувача

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

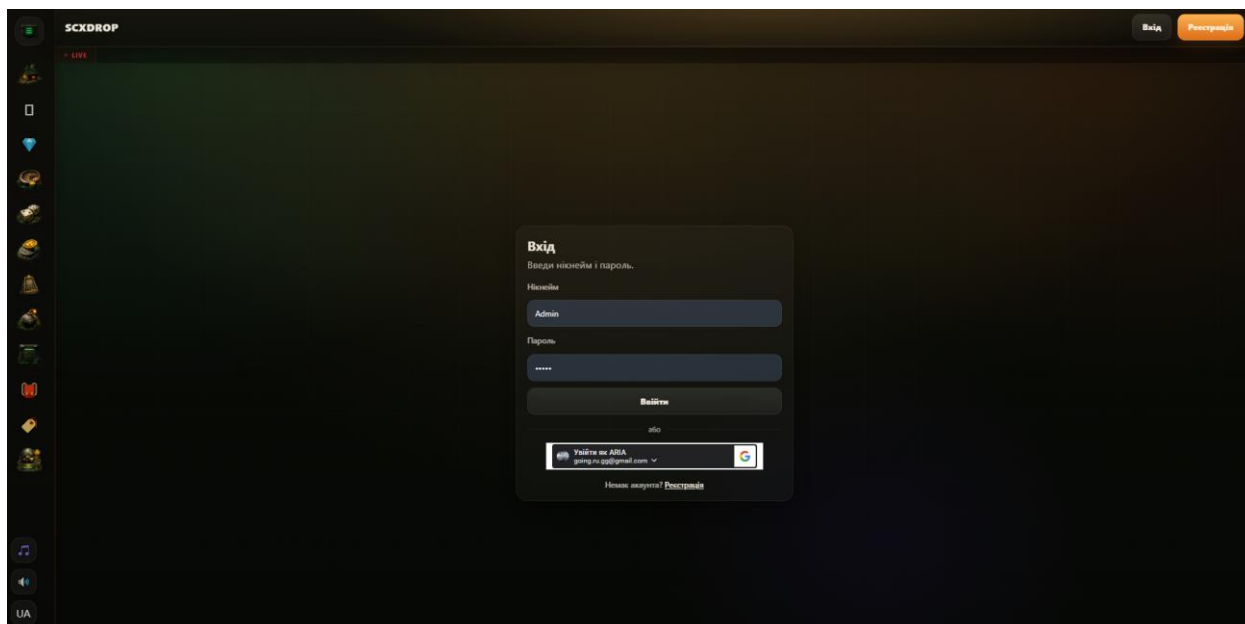


Рис. 3.4 – Сторінка входу користувача

Сторінка гри Dice (рис. 3.5) містить поле розміру ставки, повзунок порогу Roll Over та індикатори ймовірності виграшу і множника, які оновлюються миттєво під час руху повзунка. Після кнопки «Грати» відображається анімований результат кидка.

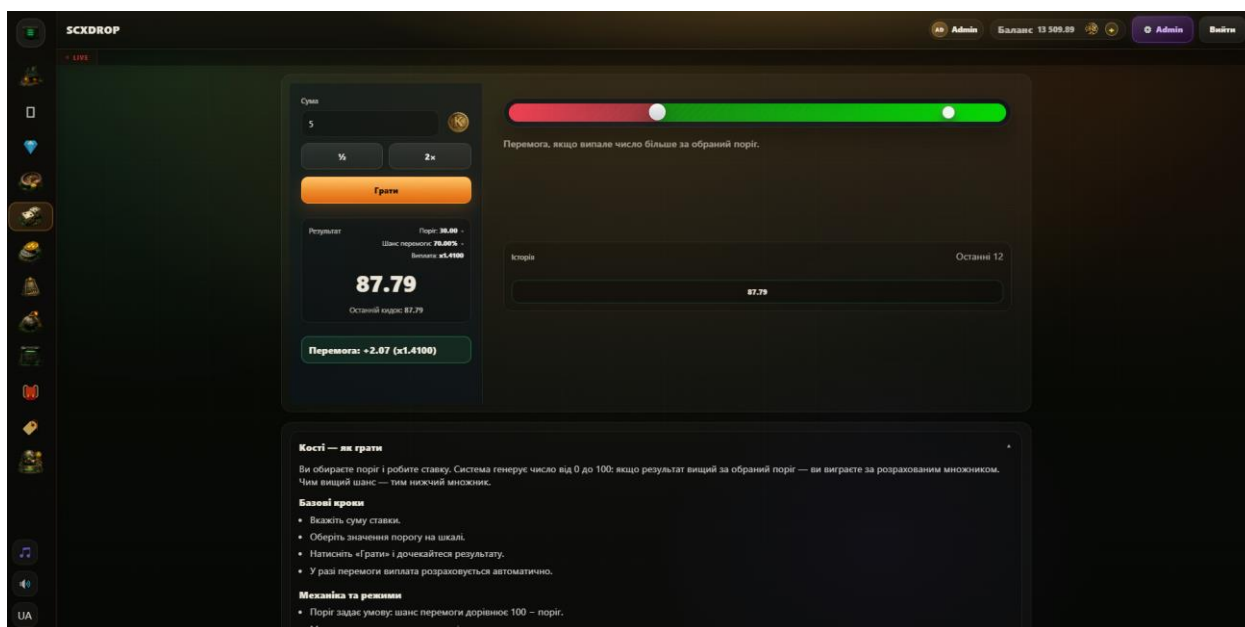


Рис. 3.5 – Сторінка гри Dice

Сторінка рулетки (рис. 3.6) відтворює класичне європейське ігрове поле: гравець розставляє фішки на числа, кольори, парність, дюжини чи колонки,

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		49

бачить сумарну ставку та запускає спін з анімацією колеса й підсвіткою переможного сектора.

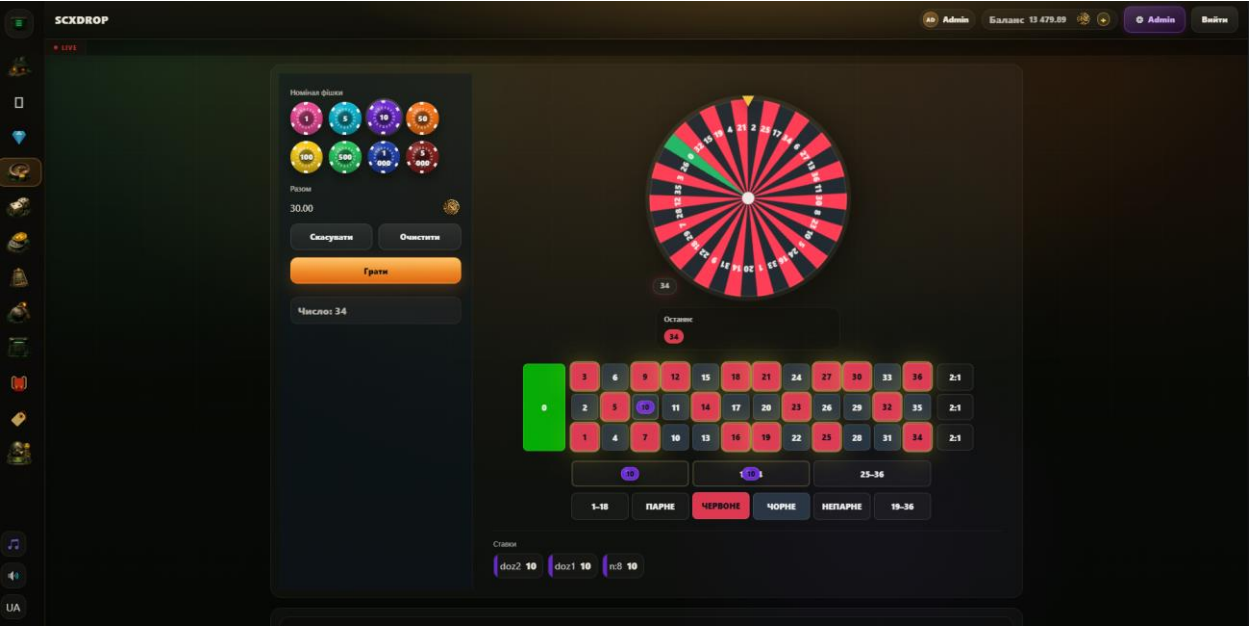


Рис. 3.6 – Сторінка гри Roulette

У грі Mines (рис. 3.7) гравець задає кількість мін, після чого відкриває клітинки поля 5×5. Поточний і наступний множники відображаються над полем; кнопка «Забрати» дозволяє завершити раунд у будь-який момент. У разі вибуху розкривається все поле з позиціями мін.

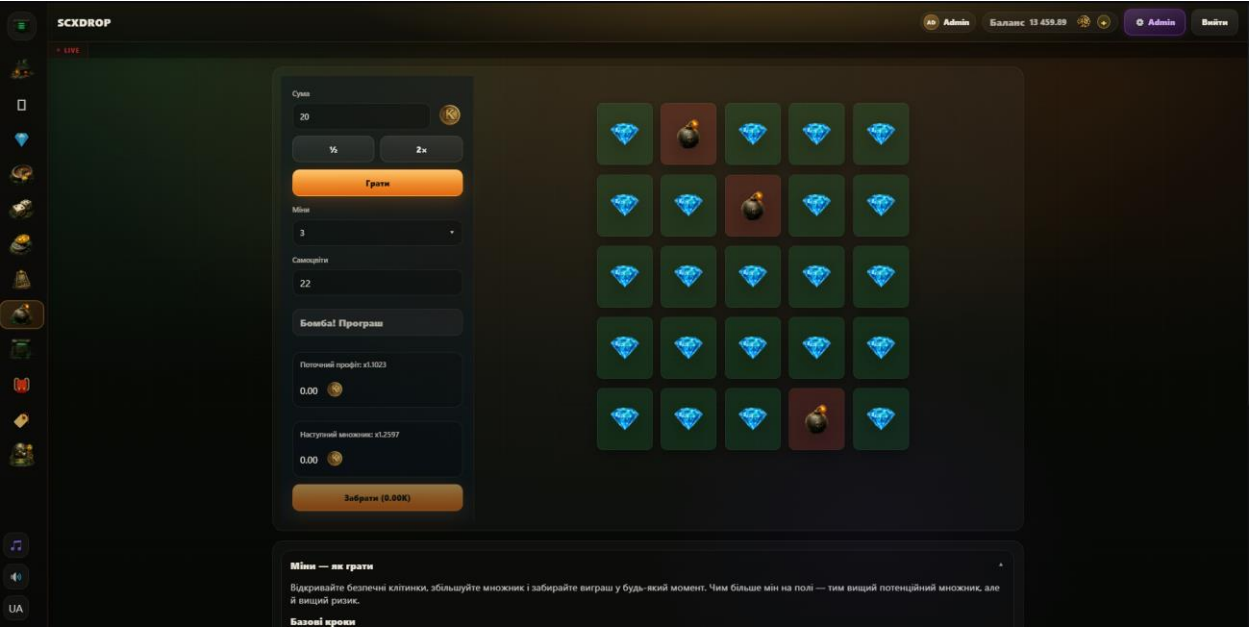
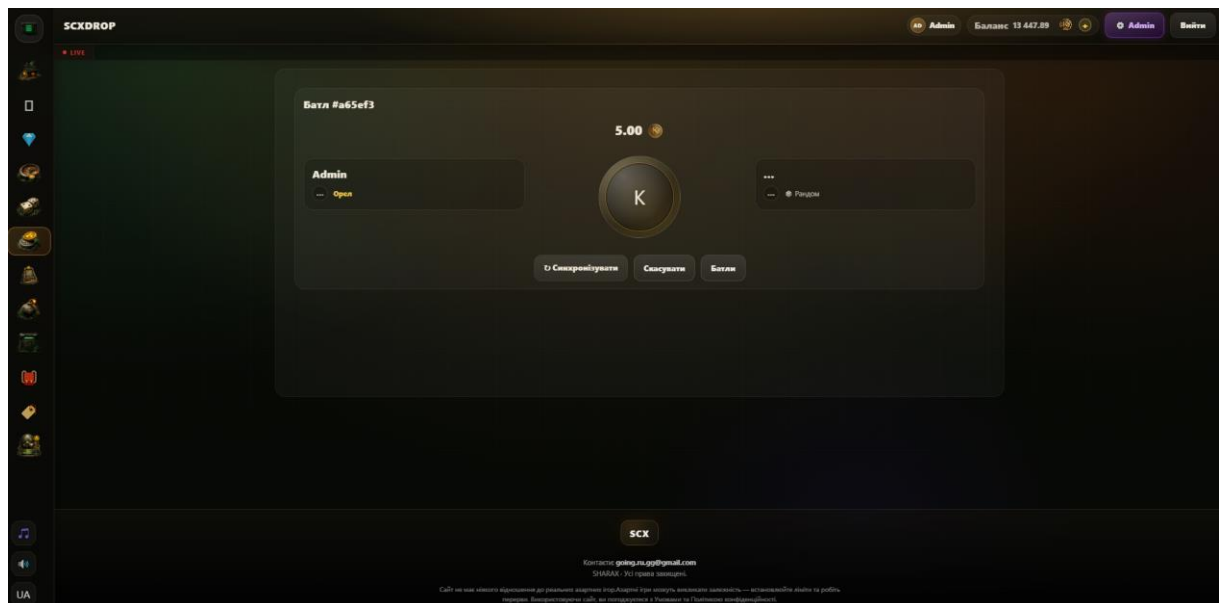


Рис. 3.7 – Сторінка гри Mines

The image shows a screenshot of the SCXDROP website interface. At the top, there's a navigation bar with the SCXDROP logo, a balance of 17,098.39, and buttons for Admin, Login, and Register. The left sidebar contains various game icons. The main content area is divided into two sections. The left section, titled 'Колесо' (Wheel), has a 'Сумма' (Sum) of 5, a 'Колесо' button, and a 'Колесо' button. Below this is a 'Складність' (Difficulty) dropdown set to 'Середня', a 'Рівень' (Level) dropdown set to '16', and a 'Задача' (Task) section with a 'Візьми' (Take) button. The right section displays a large, dark, triangular pattern of dots, resembling a Pascal's triangle, with a few dots highlighted in yellow. At the bottom, there is a row of 16 colored circles (red, orange, yellow, green, blue, purple) with numbers 1 through 16 inside them. The bottom section is titled 'Пілімо — як грати' (PiliMo — how to play) and contains text about the game mechanics.

Лобі битв Coin Flip (рис. 3.9) показує відкриті битви з нікнеймом творця, ставкою та обраною стороною монети; список оновлюється фоновно без перезавантаження сторінки. Користувач може створити власну битву, обравши ставку та сторону (або «випадкову»), чи приєднатися до наявної. Сторінка самої битви (рис. 3.10) відображає обох гравців, їхню готовність, 3-секундний відлік та анімований підкид монети з результатом.



		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

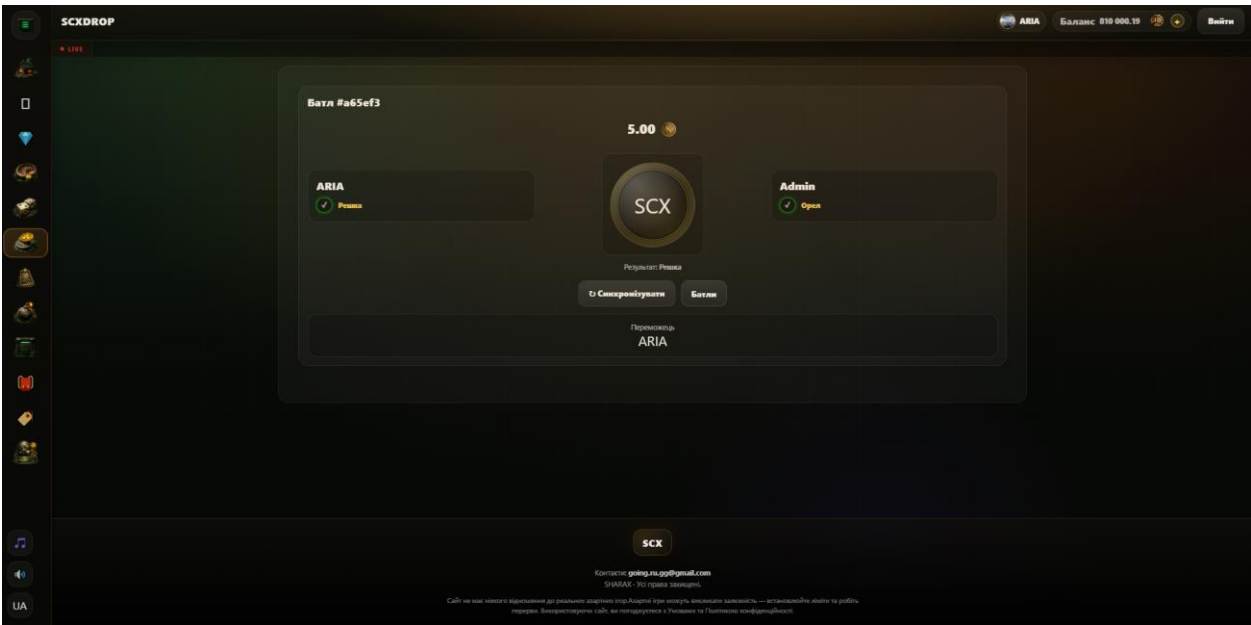


Рис. 3.10 – Сторінка битви Coin Flip

Розділ кейсів (рис. 3.11) містить три кейси з цінами та повним переліком можливих предметів і шансів. Після купівлі відтворюється рулетка-анімація прокрутки предметів, що зупиняється на випадовому предметі; предмет одразу з’являється в інвентарі.

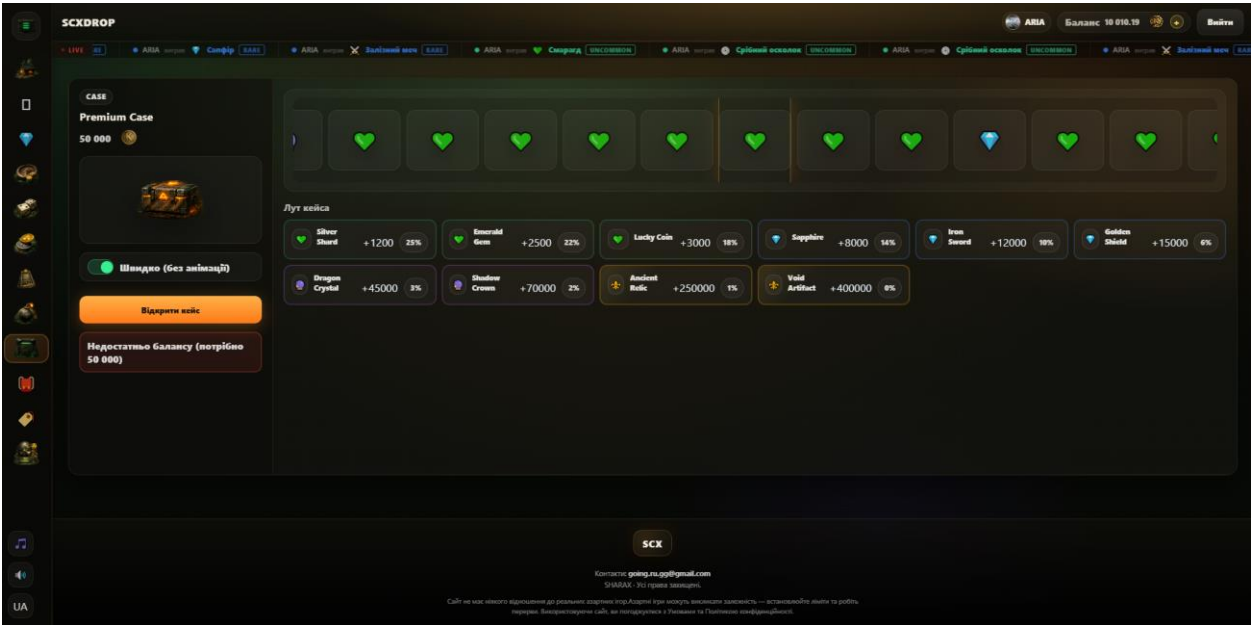


Рис. 3.11 – Сторінка кейсів та відкриття кейсу

Гра «Клікер» (рис. 3.12) відображає монету для кліків, лічильник коїнів, пасивний дохід за секунду та магазин покращень двох типів — за коїни та за

предмети з інвентаря. Окрема кнопка конвертує накопичені коїни в основну валюту К.

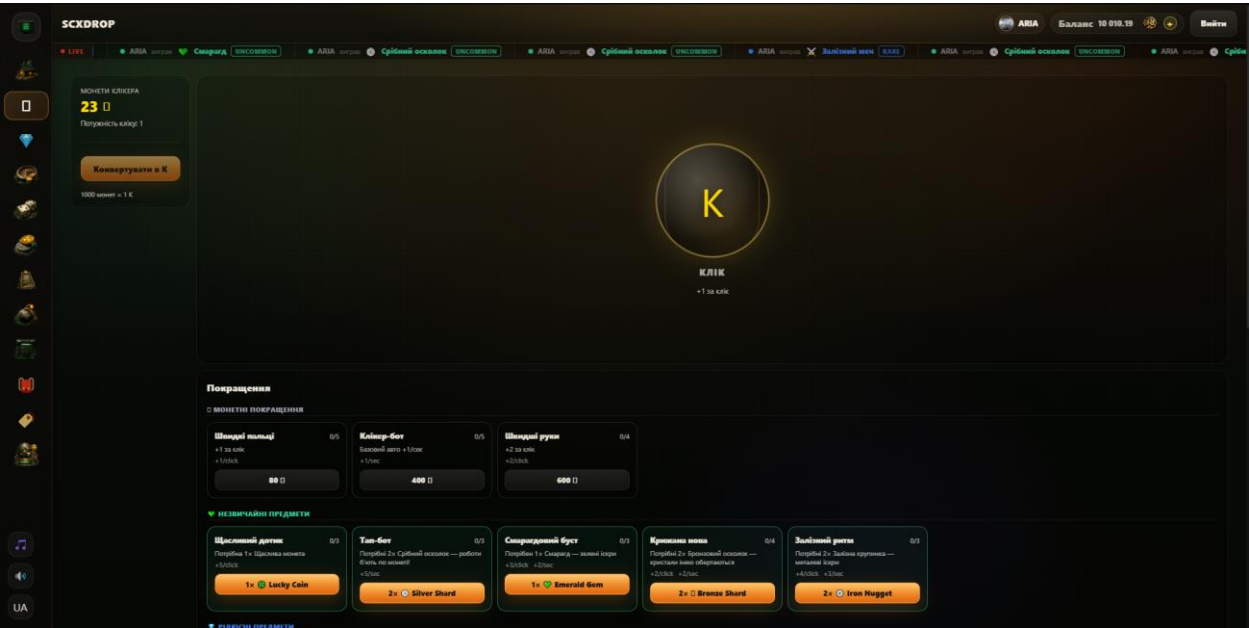


Рис. 3.12 – Сторінка гри «Клікер»

Інвентар (рис. 3.13) показує всі предмети гравця з фільтрами за рідкістю та пошуком; з картки предмета можна одразу виставити його на аукціон. Сторінка аукціону (рис. 3.14) містить список активних лотів з пошуком, фільтром за рідкістю та сортуванням за ціною; купівля підтверджується модальним вікном із зазначенням комісії платформи.

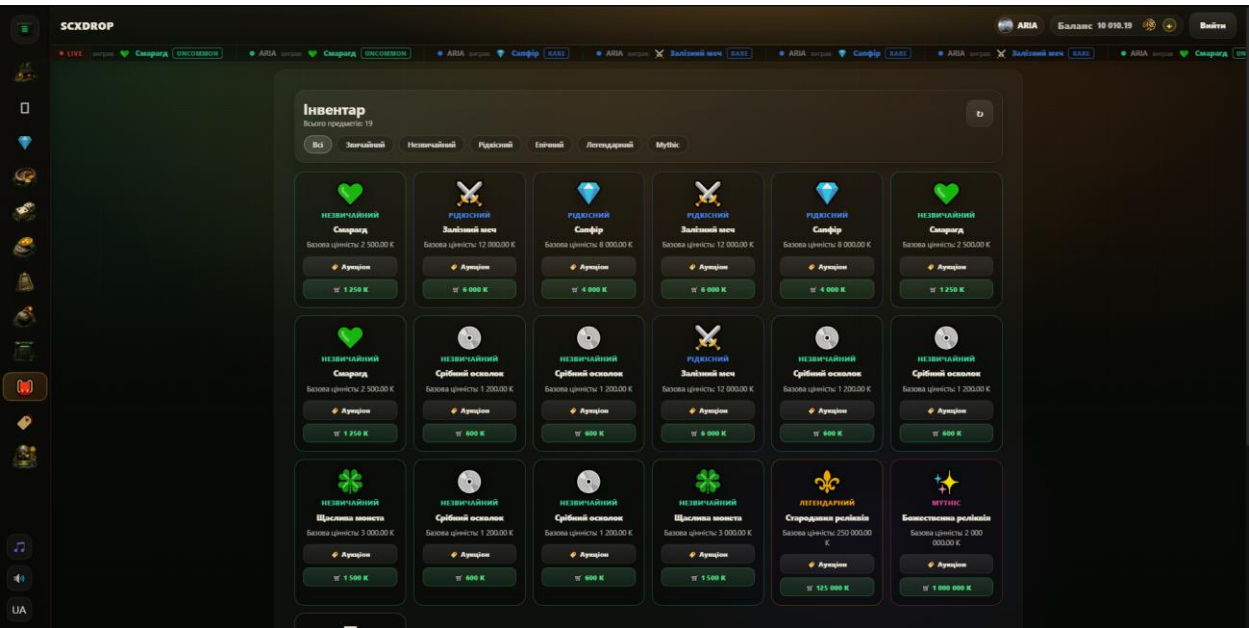


Рис. 3.13 – Сторінка інвентаря користувача

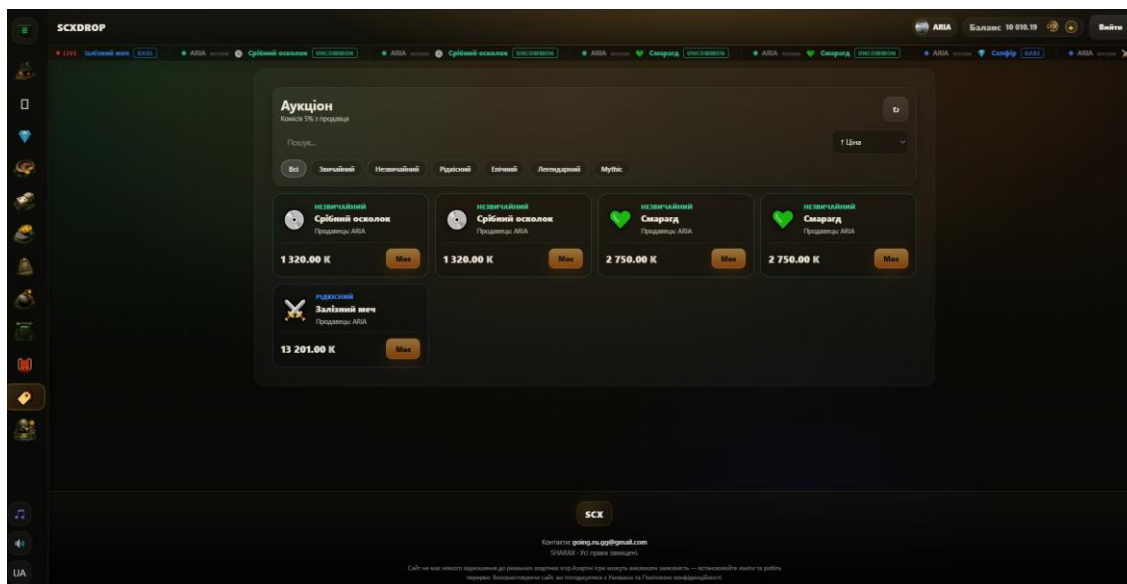


Рис. 3.14 – Сторінка аукціону

Профіль користувача (рис. 3.15) об'єднує налаштування акаунта (нікнейм, аватар), агреговану статистику за кожною грою (кількість ігор, поставлено, виграно, найбільший виграш), отримані досягнення.

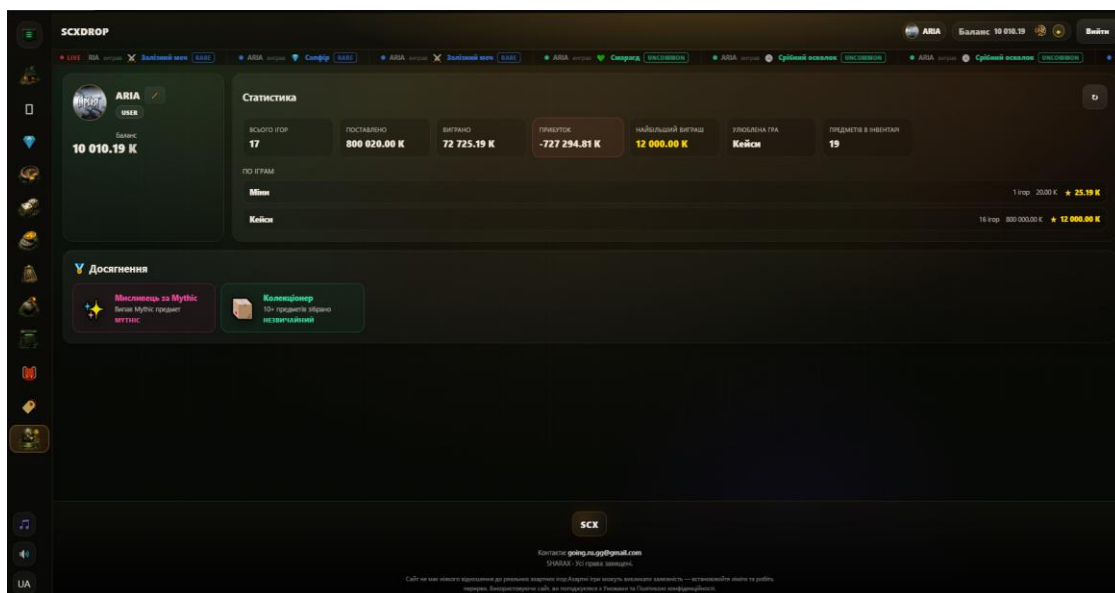


Рис. 3.15 – Сторінка профілю користувача

Адміністративна панель (рис. 3.16) доступна лише користувачам із роллю адміністратора і поділена на вкладки. Вкладка користувачів надає серверний пошук, фільтри за роллю та статусом і посторінковий перегляд, а для кожного облікового запису — детальну інформацію, коригування балансу, блокування чи розблокування з підтвердженням і видачу досягнень. Окремі

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

вкладки відображають історію битв Coin Flip, перелік поповнень із сумарним доходом, журнал дій адміністратора, а також аналітику — статистику ставок і виплат по кожній грі з показником повернення та розподіл випадінь предметів за рідкістю.

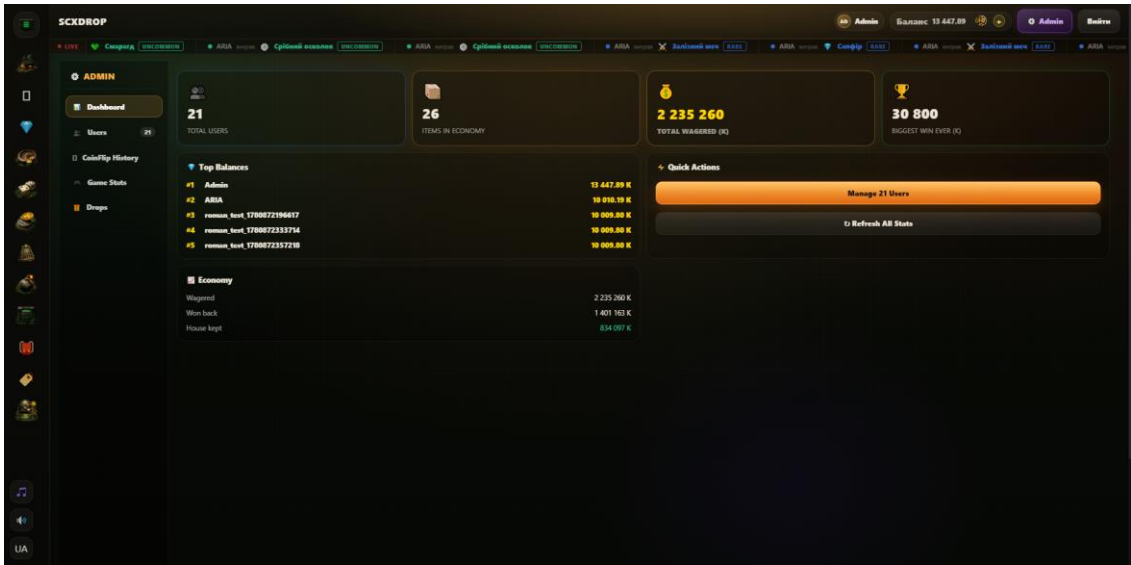


Рис. 3.16 – Адміністративна панель

Сторінка поповнення балансу (рис. 3.17) відкривається кнопкою біля показника балансу. Вона містить кілька пакетів віртуальної валюти та офіційну кнопку Google Pay; після підтвердження оплати баланс поповнюється миттєво, а нижче відображається історія поповнень користувача.

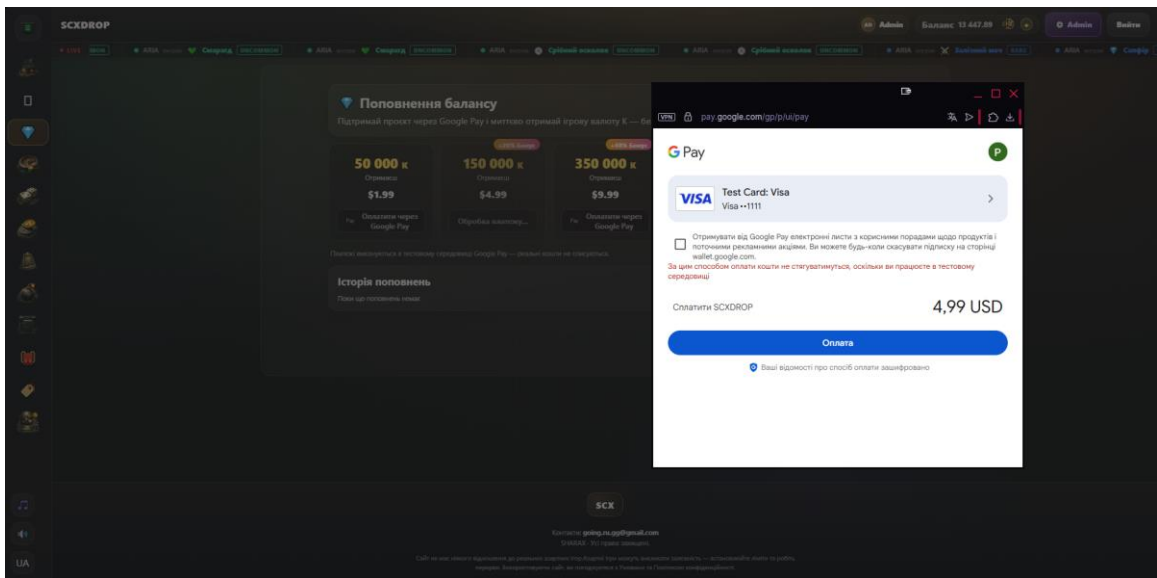


Рис. 3.17 – Сторінка поповнення балансу через Google Pay

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

3.3 Тестування роботи програмного комплексу

Якість системи забезпечується багаторівневою стратегією перевірок: статичний аналіз коду, модульні тести, наскрізні (end-to-end) тести та ручне функціональне тестування за тест-кейсами. Така багаторівнева стратегія дозволяє виявляти дефекти на різних етапах — від помилок типізації ще до запуску застосунку під час статичного аналізу до перевірки повних користувацьких сценаріїв під час наскрізного й ручного тестування.

Під час статичного аналізу вся клієнтська кодова база типізована TypeScript [28]; команда `vue-tsc --noEmit` перевіряє коректність типів у компонентах Vue. Лінер ESLint [4] із правилами для Vue та TypeScript запускається в режимі нульової толерантності до попереджень (`--max-warnings=0`); форматування уніфіковано Prettier [22]. Ці перевірки об'єднані командою `npm run check` і виконуються перед кожною збіркою.

Модульне тестування виконано фреймворком Vitest [33]. Тести покривають критичну інфраструктуру клієнта: HTTP-клієнт (коректне додавання заголовка Authorization, обробка відповідей 401, нормалізація серверних помилок), модуль обробки помилок та функції форматування чисел і валюти. Запуск — командою `npm test`; результат виконання наведено на рис. 3.18.

```
PS 6:\dyplom_work_front> npm test

> test
> vitest run

RUN v4.1.8 6:/dyplom_work_front

✓ src/utils/format.test.ts (18 tests) 7ms
✓ src/core/http/apiClient.test.ts (7 tests) 18ms
✓ src/utils/errors.test.ts (12 tests) 7ms

Test Files 3 passed (3)
Tests 37 passed (37)
Start at 12:21:58
Duration 27.85s (transform 593ms, setup 0ms, import 715ms, tests 32ms, environment 72.20s)

PS 6:\dyplom_work_front>
```

Рис. 3.18 – Результати виконання модульних тестів

Наскрізне тестування реалізовано фреймворком Playwright [20]: сценарій `tests/e2e/login.spec.ts` відтворює повний шлях користувача — відкриття сторінки входу, заповнення форми, автентифікацію та перевірку відображення профілю в шапці. Тести запускаються в реальному браузері Chromium проти запущеного застосунку. Такий сценарій підтверджує коректність інтеграції всіх ланок системи — клієнта, API-шлюзу та серверних мікросервісів — у межах єдиного реального користувацького потоку, чого не здатні гарантувати ізольовані модульні тести.

API-тестування виконувалося інструментом Postman: для кожного ендпоінта перевірено коди відповіді за коректних і некоректних даних (200/201, 400 — валідація, 401 — без токена, 403 — без прав адміністратора, 404 — відсутній ресурс, 409 — зайнятий нікнейм). Запити для всіх ендпоінтів зібрано в єдину колекцію Postman, що дозволяє повторно прогонити перевірку контракту API після кожної зміни серверної логіки й вчасно виявити порушення сумісності.

Ручне функціональне тестування свідомо охопило не лише вхід і реєстрацію, а й усі ключові грошові та ігрові операції — ставки в мінііграх, відкриття кейсів, торгівлю на аукціоні, битви Coin Flip у реальному часі, конвертацію валюти в клікері та поповнення балансу, — оскільки саме ці сценарії безпосередньо впливають на баланс користувача й вимагають коректної серверної обробки. Для кожного сценарію перевірялися як основний («щасливий») шлях, так і межові випадки — недостатній баланс, зайнятий нікнейм, доступ без прав, — що дає змогу пересвідчитися в правильній поведінці системи і за коректних, і за помилкових дій користувача.

Ручне функціональне тестування проведено за тест-кейсами, що покривають основні сценарії користувача (табл. 3.1).

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		57

Таблиця 3.1.

Тест-кейси функціонального тестування

№	Сценарій (кроки)	Очікуваний результат	Статус
1	Реєстрація нового користувача з унікальним нікнеймом	Акаунт створено, видано JWT, баланс 10 000 К, перенаправлення на головну	Пройдено
2	Реєстрація з уже зайнятим нікнеймом	Помилка 409 «Нікнейм вже зайнятий», акаунт не створено	Пройдено
3	Вхід із неправильним паролем	Помилка 401, повідомлення про невірні дані, токен не видано	Пройдено
4	Ставка в Dice за достатнього балансу	Ставка списана, результат відображено, статистика оновлена	Пройдено
5	Ставка, що перевищує баланс	Відмова «Insufficient balance», баланс не змінився	Пройдено
6	Mines: відкриття кількох клітинок і кешаут	Виграш = ставка × множник, нарахований на баланс; поле розкрито	Пройдено
7	Відкриття кейсу Starter	Вартість списана, предмет випав згідно з таблицею шансів і з'явився в інвентарі	Пройдено
8	Виставлення предмета на аукціон і купівля іншим акаунтом	Покупець отримав предмет, продавець — 95 % ціни, лот закрито	Пройдено
9	Повний цикл битви Coin Flip (два акаунти)	Стани OPEN→FULL→COUNTDOWN→FINISHED, переможець отримав подвоєну ставку, оновлення в реальному часі	Пройдено
10	Конвертація 2500 коїнів у клікери	Нараховано 2 К, залишок 500 коїнів збережено	Пройдено
11	Поповнення балансу через Google Pay (тестове середовище)	Баланс збільшено на суму пакета, створено запис донату; повторний запит з тим самим ключем не нараховує валюту вдруге	Пройдено
12	Відкриття /admin під звичайним користувачем	Перенаправлення на головну сторінку, доступ заборонено	Пройдено

Усі тест-кейси виконано успішно. Поєднання статичного аналізу, автоматизованих тестів і ручної перевірки за сценаріями підтверджує коректність роботи реалізованих алгоритмів, стабільність обміну даними між клієнтом і сервісами та відповідність системи поставленим вимогам.

ВИСНОВКИ ДО ТРЕТЬОГО РОЗДІЛУ

У третьому розділі описано порядок розгортання та роботи системи. Завдяки контейнеризації серверна частина піднімається однією командою, яка автоматично запускає шлюз, базу даних і обидва сервіси та створює схему даних; для хмарного середовища застосовано платформи Railway для бекенду й Vercel для статичного фронтенду.

Розглянуто структуру та основні елементи інтерфейсу: каталог ігор, екрани п'яти мініігор, кейси, інвентар, аукціон, клікер, профіль зі статистикою, сторінку поповнення балансу та адміністративну панель. Інтерфейс є адаптивним і доступний двома мовами.

Тестування проведено на кількох рівнях — статичний аналіз типів і лінтинг, модульні тести, наскрізний сценарій та ручні тест-кейси, що охопили основний функціонал: реєстрацію, ігри, економіку предметів, битви в реальному часі та адміністрування. Усі перевірки пройдено успішно, що підтверджує коректність роботи реалізованих алгоритмів.

Таким чином, третій розділ підтвердив працездатність системи та її готовність до використання за призначенням.

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

ВИСНОВКИ

У випускній кваліфікаційній роботі бакалавра розв’язано задачу створення вебплатформи симулятора мініігор — інформаційної системи з казино-подібними ігровими механіками та віртуальною валютою без використання реальних коштів. Усі завдання, поставлені у вступі, виконано повністю.

У першому розділі проаналізовано предметну область та чотири існуючі ігрові платформи (GGBET, BETON, Gorilla, 777.ua). Виявлено типові недоліки аналогів — перевантаженість промоматеріалами, непрозорість ігрової логіки та слабку персональну аналітику, — які враховано під час проєктування власного рішення. Обґрунтовано клієнт-серверну архітектуру з мікросервісним бекендом і технологічний стек: Vue 3 + Vite на клієнті, Node.js + Express на сервері, PostgreSQL як сховище даних та Docker для уніфікації середовища.

У другому розділі спроектовано систему: визначено акторів і варіанти використання, розроблено базу даних із дев’яти таблиць з точними грошовими типами та JSONB-структурами, формалізовано алгоритми семи ігрових механік (Dice, Roulette, Mines, Plinko, Coin Flip, кейси, клікер) та економіки предметів з аукціоном. Реалізовано два мікросервіси Node.js за єдиним API-шлюзом nginx, захист JWT-токенами та роллю адміністратора з перевіркою в базі даних. Критичні грошові операції виконуються атомарними умовними запитами та транзакціями, що унеможливорює від’ємні баланси і подвійні виплати; події битв доставляються гравцям у реальному часі через Server-Sent Events.

У третьому розділі описано розгортання системи (одна команда docker compose локально; Railway + Vercel у хмарі), шістнадцять сторінок інтерфейсу та результати тестування на чотирьох рівнях — від статичного аналізу TypeScript і ESLint до модульних тестів Vitest, наскрізного сценарію Playwright та дванадцяти ручних тест-кейсів. Усі перевірки пройдено успішно.

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		60

Результатом роботи є повнофункціональна вебплатформа, розгорнута у хмарному середовищі: користувачі реєструються (у тому числі через Google), грають у сім мініігор, накопичують предмети шести рівнів рідкості, торгують ними на аукціоні та відстежують власну статистику і досягнення; адміністратор керує користувачами та балансами, а також переглядає аналітику ігор і дропів. Практичне значення полягає у створенні безпечного розважального сервісу без фінансових ризиків, який водночас демонструє промислові підходи до побудови мікросервісних систем із віртуальною економікою.

Перспективи подальшого розвитку платформи: впровадження криптографічно перевірюваної чесності результатів (provably fair) на основі пар seed-значень; перехід на WebSocket для двонаправленої взаємодії в іграх «гравець проти гравця»; глобальні рейтинги та турніри; мобільний застосунок або PWA; розширення колекції мініігор завдяки модульній архітектурі ігрових маршрутів.

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Newman S. Building Microservices: Designing Fine-Grained Systems / S. Newman. 2nd ed. Sebastopol : O'Reilly Media, 2021. 612 p.
2. Docker Inc. Docker Compose Documentation : електронний ресурс. URL: <https://docs.docker.com/compose/> (дата звернення: 12.06.2026).
3. Docker Inc. Docker Documentation : електронний ресурс. URL: <https://docs.docker.com/> (дата звернення: 12.06.2026).
4. OpenJS Foundation. ESLint Documentation : електронний ресурс. URL: <https://eslint.org/docs/latest/> (дата звернення: 11.06.2026).
5. OpenJS Foundation. Express — fast, unopinionated, minimalist web framework for Node.js : електронний ресурс. URL: <https://expressjs.com/> (дата звернення: 11.06.2026).
6. MDN Web Docs. Fetch API : електронний ресурс. URL: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API (дата звернення: 14.06.2026).
7. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : PhD dissertation / R. T. Fielding. Irvine : University of California, 2000. 162 p.
8. Fowler M. Microservices: a definition of this new architectural term : електронний ресурс / M. Fowler. 2014. URL: <https://martinfowler.com/articles/microservices.html> (дата звернення: 13.06.2026).
9. Google LLC. Sign In With Google for Web : електронний ресурс. URL: <https://developers.google.com/identity/gsi/web> (дата звернення: 14.06.2026).
10. Knuth D. E. The Art of Computer Programming. Vol. 2 : Seminumerical Algorithms / D. E. Knuth. 3rd ed. Boston : Addison-Wesley, 1997. 762 p.

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

- 11.Mock Service Worker contributors. Mock Service Worker Documentation :
електронний ресурс. URL: <https://mswjs.io/docs> (дата звернення:
10.06.2026).
- 12.Nginx Inc. nginx Documentation : електронний ресурс. URL:
<https://nginx.org/en/docs/> (дата звернення: 12.06.2026).
- 13.Casciaro M. Node.js Design Patterns / М. Casciaro, L. Mammino. 3rd ed.
Birmingham : Packt Publishing, 2020. 664 p.
- 14.OpenJS Foundation. Node.js v22 Documentation : електронний ресурс.
URL: <https://nodejs.org/docs/latest-v22.x/api/> (дата звернення: 11.06.2026).
- 15.node-postgres contributors. node-postgres Documentation : електронний
ресурс. URL: <https://node-postgres.com/> (дата звернення: 12.06.2026).
- 16.Object Management Group. OMG Unified Modeling Language (UML).
Version 2.5.1 : електронний ресурс. 2017. URL:
<https://www.omg.org/spec/UML/2.5.1/> (дата звернення: 13.06.2026).
- 17.OWASP Foundation. OWASP Authentication Cheat Sheet : електронний
ресурс. URL:
https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html
(дата звернення: 13.06.2026).
- 18.OWASP Foundation. OWASP Top 10:2021 : електронний ресурс. URL:
<https://owasp.org/Top10/> (дата звернення: 13.06.2026).
- 19.Pinia contributors. Pinia Documentation : електронний ресурс. URL:
<https://pinia.vuejs.org/> (дата звернення: 11.06.2026).
- 20.Microsoft Corporation. Playwright Documentation : електронний ресурс.
URL: <https://playwright.dev/docs/intro> (дата звернення: 13.06.2026).
- 21.PostgreSQL Global Development Group. PostgreSQL 15 Documentation :
електронний ресурс. URL: <https://www.postgresql.org/docs/15/> (дата
звернення: 12.06.2026).
- 22.Prettier contributors. Prettier Documentation : електронний ресурс. URL:
<https://prettier.io/docs/> (дата звернення: 11.06.2026).

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

- 23.Provos N. A Future-Adaptable Password Scheme / N. Provos, D. Mazières // Proceedings of the FREENIX Track: 1999 USENIX Annual Technical Conference. Monterey, 1999. P. 81–91.
- 24.Railway Corp. Railway Documentation : електронний ресурс. URL: <https://docs.railway.com/> (дата звернення: 14.06.2026).
- 25.Hardt D. RFC 6749. The OAuth 2.0 Authorization Framework : електронний ресурс / D. Hardt. IETF, 2012. URL: <https://datatracker.ietf.org/doc/html/rfc6749> (дата звернення: 14.06.2026).
- 26.Jones M. RFC 7519. JSON Web Token (JWT) : електронний ресурс / M. Jones, J. Bradley, N. Sakimura. IETF, 2015. URL: <https://datatracker.ietf.org/doc/html/rfc7519> (дата звернення: 14.06.2026).
- 27.MDN Web Docs. Server-Sent Events : електронний ресурс. URL: https://developer.mozilla.org/en-US/docs/Web/API/Server-sent_events (дата звернення: 14.06.2026).
- 28.Microsoft Corporation. TypeScript Documentation : електронний ресурс. URL: <https://www.typescriptlang.org/docs/> (дата звернення: 11.06.2026).
- 29.UnJS contributors. Unhead Documentation : електронний ресурс. URL: <https://unhead.unjs.io/> (дата звернення: 11.06.2026).
- 30.Vercel Inc. Vercel Documentation : електронний ресурс. URL: <https://vercel.com/docs> (дата звернення: 11.06.2026).
- 31.Vite team. Vite Documentation : електронний ресурс. URL: <https://vite.dev/guide/> (дата звернення: 11.06.2026).
- 32.antfu-collective. vite-ssg: Static Site Generation for Vue 3 on Vite : електронний ресурс. URL: <https://github.com/antfu-collective/vite-ssg> (дата звернення: 11.06.2026).
- 33.Vitest team. Vitest Documentation : електронний ресурс. URL: <https://vitest.dev/guide/> (дата звернення: 11.06.2026).
- 34.Intlify. Vue I18n Documentation : електронний ресурс. URL: <https://vue-i18n.intlify.dev/> (дата звернення: 11.06.2026).

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

35.Vue.js. Vue Router Documentation : електронний ресурс. URL:
<https://router.vuejs.org/> (дата звернення: 14.06.2026).

36.Vue.js. Vue.js Guide : електронний ресурс. URL:
<https://vuejs.org/guide/introduction.html> (дата звернення: 14.06.2026).

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		65

ДОДАТКИ

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

Лістинг коду файлу/db.js

```

import pg from 'pg';
const { Pool } = pg;
const pool = new Pool({ connectionString: process.env.DATABASE_URL });

export async function initDb() {
  await pool.query(`
    CREATE TABLE IF NOT EXISTS mines_sessions (
      id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
      user_id UUID NOT NULL UNIQUE,
      bet NUMERIC(20,2) NOT NULL,
      mines_count INT NOT NULL,
      mine_positions JSONB NOT NULL,
      opened JSONB NOT NULL DEFAULT '[]',
      status VARCHAR(20) NOT NULL DEFAULT 'ACTIVE',
      created_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
    );

    CREATE TABLE IF NOT EXISTS battles (
      id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
      amount NUMERIC(20,2) NOT NULL,
      status VARCHAR(20) NOT NULL DEFAULT 'OPEN',
      creator_id UUID NOT NULL,
      creator_nick VARCHAR(100) NOT NULL,
      creator_side VARCHAR(10),
      creator_ready BOOLEAN NOT NULL DEFAULT FALSE,
      joiner_id UUID,
      joiner_nick VARCHAR(100),
      joiner_side VARCHAR(10),
      joiner_ready BOOLEAN NOT NULL DEFAULT FALSE,
      countdown_started_at TIMESTAMPTZ,
      winner_id UUID,
      winner_side VARCHAR(10),
      result_side VARCHAR(10),
      created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
      updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
    );

    CREATE TABLE IF NOT EXISTS items (
      id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
      user_id UUID NOT NULL,
      item_def_id VARCHAR(50) NOT NULL,
      source VARCHAR(30) NOT NULL DEFAULT 'case',
      created_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
    );

    CREATE TABLE IF NOT EXISTS auction_listings (
      id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
      seller_id UUID NOT NULL,
      seller_nick VARCHAR(100) NOT NULL,
      item_id UUID NOT NULL REFERENCES items(id) ON DELETE CASCADE,
      item_def_id VARCHAR(50) NOT NULL,
      price NUMERIC(20,2) NOT NULL,
      status VARCHAR(20) NOT NULL DEFAULT 'ACTIVE',
      buyer_id UUID,
      buyer_nick VARCHAR(100),
  
```

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

```

        created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
        sold_at TIMESTAMPTZ
    );

CREATE TABLE IF NOT EXISTS clicker_state (
    user_id UUID PRIMARY KEY,
    coins BIGINT NOT NULL DEFAULT 0,
    click_power INT NOT NULL DEFAULT 1,
    auto_power INT NOT NULL DEFAULT 0,
    upgrades JSONB NOT NULL DEFAULT '{}',
    last_auto_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

CREATE TABLE IF NOT EXISTS game_stats (
    user_id UUID NOT NULL,
    game_type VARCHAR(30) NOT NULL,
    games_played INT NOT NULL DEFAULT 0,
    total_wagered NUMERIC(20,2) NOT NULL DEFAULT 0,
    total_won NUMERIC(20,2) NOT NULL DEFAULT 0,
    biggest_win NUMERIC(20,2) NOT NULL DEFAULT 0,
    PRIMARY KEY (user_id, game_type)
);
`);
}

export default pool;

```

Лістинг коду файлу /routes/battles.js

```

import { Router } from 'express';
import pool from '../db.js';
import { requireAuth, addBalance, deductBalance } from
'../middleware/auth.js';
import { ITEM_DEFS } from '../itemDefs.js';

const router = Router();

function listingDto(row) {
    const def = ITEM_DEFS[row.item_def_id] ?? {};
    return {
        id: row.id,
        sellerId: row.seller_id,
        sellerNick: row.seller_nick,
        itemId: row.item_id,
        itemDefId: row.item_def_id,
        name: def.name ?? row.item_def_id,
        nameUa: def.nameUa ?? row.item_def_id,
        rarity: def.rarity ?? 'common',
        icon: def.icon ?? '🎁',
        color: def.color ?? '#9ca3af',
        baseValue: def.value ?? 0,
        price: parseFloat(row.price),
        status: row.status,
        buyerId: row.buyer_id ?? null,
        buyerNick: row.buyer_nick ?? null,
        createdAt: row.created_at,
        soldAt: row.sold_at ?? null,
    };
}

```

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		68

```

// GET /api/v1/auction - list active listings
router.get('/', async (req, res) => {
  try {
    const { rows } = await pool.query(
      `SELECT * FROM auction_listings WHERE status = 'ACTIVE' ORDER BY
created_at DESC LIMIT 100`
    );
    res.json({ ok: true, listings: rows.map(listingDto) });
  } catch (err) {
    console.error(err);
    res.status(500).json({ ok: false, error: 'Internal server error' });
  }
});

// POST /api/v1/auction - list an item
router.post('/', requireAuth, async (req, res) => {
  const { itemId, price } = req.body;
  const p = Number(price);
  if (!Number.isFinite(p) || p <= 0) {
    return res.status(400).json({ ok: false, error: 'Invalid price' });
  }
  const client = await pool.connect();
  try {
    await client.query('BEGIN');
    // Verify item belongs to user and is not already listed
    const { rows: itemRows } = await client.query(
      `SELECT i.* FROM items i
      LEFT JOIN auction_listings al ON al.item_id = i.id AND al.status =
'ACTIVE'
      WHERE i.id = $1 AND i.user_id = $2 AND al.id IS NULL`,
      [itemId, req.user.id]
    );
    if (!itemRows[0]) {
      await client.query('ROLLBACK');
      return res.status(404).json({ ok: false, error: 'Item not found or
already listed' });
    }
    const item = itemRows[0];
    const { rows } = await client.query(
      `INSERT INTO auction_listings (seller_id, seller_nick, item_id,
item_def_id, price)
      VALUES ($1, $2, $3, $4, $5) RETURNING *`,
      [req.user.id, req.user.nickname, itemId, item.item_def_id, p]
    );
    await client.query('COMMIT');
    res.status(201).json({ ok: true, listing: listingDto(rows[0]) });
  } catch (err) {
    await client.query('ROLLBACK');
    console.error(err);
    res.status(500).json({ ok: false, error: 'Internal server error' });
  } finally {
    client.release();
  }
});

// DELETE /api/v1/auction/:id - delist
router.delete('/:id', requireAuth, async (req, res) => {
  try {
    const { rows } = await pool.query(
      `UPDATE auction_listings SET status = 'CANCELLED'
      WHERE id = $1 AND seller_id = $2 AND status = 'ACTIVE' RETURNING *`,
      [req.params.id, req.user.id]
    );
  }
});

```

```

    );
    if (!rows[0]) return res.status(404).json({ ok: false, error: 'Listing
not found' });
    res.json({ ok: true });
  } catch (err) {
    console.error(err);
    res.status(500).json({ ok: false, error: 'Internal server error' });
  }
});

// POST /api/v1/auction/:id/buy
router.post('/:id/buy', requireAuth, async (req, res) => {
  const client = await pool.connect();
  try {
    await client.query('BEGIN');
    const { rows: listRows } = await client.query(
      `SELECT * FROM auction_listings WHERE id = $1 AND status = 'ACTIVE'`,
      [req.params.id]
    );
    const listing = listRows[0];
    if (!listing) {
      await client.query('ROLLBACK');
      return res.status(404).json({ ok: false, error: 'Listing not found or
sold' });
    }
    if (listing.seller_id === req.user.id) {
      await client.query('ROLLBACK');
      return res.status(400).json({ ok: false, error: 'Cannot buy your own
listing' });
    }

    const price = parseFloat(listing.price);
    // Deduct from buyer
    const { rows: balRows } = await client.query(
      `UPDATE users SET balance = balance - $1 WHERE id = $2 AND balance >=
$1 RETURNING balance`,
      [price, req.user.id]
    );
    if (!balRows[0]) {
      await client.query('ROLLBACK');
      return res.status(400).json({ ok: false, error: 'Insufficient balance'
});
    }

    // Pay seller (95% - 5% marketplace fee)
    const sellerPayout = parseFloat((price * 0.95).toFixed(2));
    await client.query(
      `UPDATE users SET balance = balance + $1 WHERE id = $2`,
      [sellerPayout, listing.seller_id]
    );

    // Transfer item to buyer
    await client.query(
      `UPDATE items SET user_id = $1 WHERE id = $2`,
      [req.user.id, listing.item_id]
    );

    // Close listing
    await client.query(
      `UPDATE auction_listings SET status = 'SOLD', buyer_id = $1, buyer_nick
= $2, sold_at = NOW()
      WHERE id = $3`,

```

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		70

```

    [req.user.id, req.user.nickname, listing.id]
  );

  await client.query('COMMIT');
  res.json({ ok: true });
} catch (err) {
  await client.query('ROLLBACK');
  console.error(err);
  res.status(500).json({ ok: false, error: 'Internal server error' });
} finally {
  client.release();
}
});

export default router;

```

Лістинг коду файлу /routes/cases.js

```

import { Router } from 'express';
import pool from '../db.js';
import { requireAuth, deductBalance, addBalance, recordStats } from
'../middleware/auth.js';
import { ITEM_DEFS } from '../itemDefs.js';
import { pushDrop } from './drops.js';

const router = Router();

const CASES = {
  starter: {
    id: 'starter', name: 'Starter Case', nameUa: 'Стартовий кейс',
    price: 10000,
    modelEmoji: '📦',
    items: [
      { itemDefId: 'wood_chip', chance: 0.30 },
      { itemDefId: 'bronze_shard', chance: 0.28 },
      { itemDefId: 'iron_nugget', chance: 0.20 },
      { itemDefId: 'silver_shard', chance: 0.12 },
      { itemDefId: 'emerald_gem', chance: 0.07 },
      { itemDefId: 'lucky_coin', chance: 0.025 },
      { itemDefId: 'sapphire', chance: 0.005 },
    ],
  },
  premium: {
    id: 'premium', name: 'Premium Case', nameUa: 'Преміум кейс',
    price: 50000,
    modelEmoji: '👛',
    items: [
      { itemDefId: 'silver_shard', chance: 0.25 },
      { itemDefId: 'emerald_gem', chance: 0.22 },
      { itemDefId: 'lucky_coin', chance: 0.18 },
      { itemDefId: 'sapphire', chance: 0.14 },
      { itemDefId: 'iron_sword', chance: 0.10 },
      { itemDefId: 'golden_shield', chance: 0.06 },
      { itemDefId: 'dragon_crystal', chance: 0.03 },
      { itemDefId: 'shadow_crown', chance: 0.018 },
      { itemDefId: 'ancient_relic', chance: 0.01 },
      { itemDefId: 'void_artifact', chance: 0.002 },
    ],
  },
  elite: {

```

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		71

```

    id: 'elite', name: 'Elite Case', nameUa: 'Елітний кейс',
    price: 100000,
    modelEmoji: '🛡️',
    items: [
      { itemDefId: 'sapphire',      chance: 0.20 },
      { itemDefId: 'iron_sword',    chance: 0.18 },
      { itemDefId: 'golden_shield', chance: 0.15 },
      { itemDefId: 'dragon_crystal', chance: 0.14 },
      { itemDefId: 'shadow_crown',  chance: 0.12 },
      { itemDefId: 'void_essence',  chance: 0.10 },
      { itemDefId: 'ancient_relic', chance: 0.055 },
      { itemDefId: 'void_artifact', chance: 0.025 },
      { itemDefId: 'phoenix_feather', chance: 0.009 },
      { itemDefId: 'divine_relic',   chance: 0.0009 },
      { itemDefId: 'cosmos_gem',     chance: 0.0001 },
    ],
  },
};

function rollItem(items) {
  const r = Math.random();
  let cumulative = 0;
  for (const item of items) {
    cumulative += item.chance;
    if (r <= cumulative) return item;
  }
  return items[items.length - 1];
}

function caseToDto(c) {
  return {
    id: c.id, name: c.name, nameUa: c.nameUa,
    price: c.price, modelEmoji: c.modelEmoji,
    items: c.items.map(i => {
      const def = ITEM_DEFS[i.itemDefId];
      return {
        itemDefId: i.itemDefId,
        name: def?.name ?? i.itemDefId,
        nameUa: def?.nameUa ?? i.itemDefId,
        rarity: def?.rarity ?? 'common',
        icon: def?.icon ?? '🛡️',
        color: def?.color ?? '#fff',
        value: def?.value ?? 0,
        chance: parseFloat((i.chance * 100).toFixed(4)), // send as
percentage (game.ts divides by 100)
        prize: def?.value ?? 0,
      };
    }),
  };
}

// GET /api/v1/games/cases
router.get('/', (req, res) => {
  res.json(Object.values(CASES).map(caseToDto));
});

// GET /api/v1/games/cases/:id
router.get('/:id', (req, res) => {
  if (req.params.id === 'game') return res.status(404).json({ ok: false,
error: 'Not found' });
  const c = CASES[req.params.id];
  if (!c) return res.status(404).json({ ok: false, error: 'Case not found'
});
});

```

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

```

    res.json(caseToDto(c));
  });

// POST /api/v1/games/cases/game/play
router.post('/game/play', requireAuth, async (req, res) => {
  const { type } = req.body;
  const c = CASES[type];
  if (!c) return res.status(400).json({ ok: false, error: 'Unknown case type'
});

  try {
    await deductBalance(req.user.id, c.price);
  } catch {
    return res.status(400).json({ ok: false, error: 'Insufficient balance'
});
  }

  const rolled = rollItem(c.items);
  const def = ITEM_DEFS[rolled.itemDefId];
  const payout = def?.value ?? 0;

  // Save item to user inventory
  const { rows: itemRows } = await pool.query(
    `INSERT INTO items (user_id, item_def_id, source) VALUES ($1, $2, 'case')
RETURNING id`,
    [req.user.id, rolled.itemDefId]
  );
  const itemId = itemRows[0]?.id;

  recordStats(req.user.id, 'cases', c.price, payout);

  // Broadcast to drops ticker
  pushDrop({
    nick: req.user.nickname,
    caseId: type,
    caseName: c.name,
    itemDefId: rolled.itemDefId,
    name: def?.name ?? rolled.itemDefId,
    nameUa: def?.nameUa ?? rolled.itemDefId,
    rarity: def?.rarity ?? 'common',
    icon: def?.icon ?? '🎁',
    color: def?.color ?? '#9ca3af',
    ts: Date.now(),
  });

  res.json({
    item: def?.name ?? rolled.itemDefId,
    itemUa: def?.nameUa ?? rolled.itemDefId,
    itemDefId: rolled.itemDefId,
    itemId,
    payout,
    rarity: def?.rarity ?? 'common',
    icon: def?.icon ?? '🎁',
    color: def?.color ?? '#fff',
  });
});

export default router;

```

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

Лістинг коду файлу /routes/clicker.js

```
import { Router } from 'express';
import pool from '../db.js';
import { requireAuth, addBalance } from '../middleware/auth.js';
import { ITEM_DEFS } from '../itemDefs.js';

const router = Router();

export const UPGRADES = [
  // — Tier 1: Coin upgrades —————
  {
    id: 'click_c1', tier: 1, visual: null,
    name: 'Quick Fingers', nameUa: 'Швидкі пальці',
    desc: '+1 per click', descUa: '+1 за клік',
    coinCost: 80, itemReq: null,
    clickBonus: 1, autoBonus: 0, maxLevel: 5,
  },
  {
    id: 'auto_c1', tier: 1, visual: null,
    name: 'Clicker Bot', nameUa: 'Кліккер-бот',
    desc: 'Basic auto +1/sec', descUa: 'Базовий авто +1/сек',
    coinCost: 400, itemReq: null,
    clickBonus: 0, autoBonus: 1, maxLevel: 5,
  },
  {
    id: 'click_c2', tier: 1, visual: null,
    name: 'Faster Hands', nameUa: 'Швидші руки',
    desc: '+2 per click', descUa: '+2 за клік',
    coinCost: 600, itemReq: null,
    clickBonus: 2, autoBonus: 0, maxLevel: 4,
  },
  // — Tier 2: Uncommon items —————
  {
    id: 'lucky_touch', tier: 2, visual: 'lucky',
    name: 'Lucky Touch', nameUa: 'Щасливий дотик',
    desc: 'Requires 1x Lucky Coin', descUa: 'Потрібна 1x Щаслива монета',
    coinCost: 0, itemReq: { itemDefId: 'lucky_coin', count: 1 },
    clickBonus: 5, autoBonus: 0, maxLevel: 3,
  },
  {
    id: 'tap_bot', tier: 2, visual: 'robot',
    name: 'Tap Bot', nameUa: 'Тап-бот',
    desc: 'Requires 2x Silver Shard — robots hit the coin!', descUa:
    'Потрібні 2x Срібний осколок — роботи б\ють по монеті!',
    coinCost: 0, itemReq: { itemDefId: 'silver_shard', count: 2 },
    clickBonus: 0, autoBonus: 5, maxLevel: 3,
  },
  {
    id: 'emerald_boost', tier: 2, visual: 'emerald',
    name: 'Emerald Boost', nameUa: 'Смарагдовий буст',
    desc: 'Requires 1x Emerald Gem — green sparkles', descUa: 'Потрібен 1x
    Смарагд — зелені іскри',
    coinCost: 0, itemReq: { itemDefId: 'emerald_gem', count: 1 },
    clickBonus: 3, autoBonus: 2, maxLevel: 3,
  },
  {
    id: 'ice_nova', tier: 2, visual: 'ice',
    name: 'Ice Nova', nameUa: 'Крижана нова',

```

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		74

```

    desc: 'Requires 2x Bronze Shard – frost crystals orbit', descUa:
'Потрібні 2× Бронзовий осколок – кристали інею обертаються',
    coinCost: 0, itemReq: { itemDefId: 'bronze_shard', count: 2 },
    clickBonus: 2, autoBonus: 2, maxLevel: 4,
  },
  {
    id: 'iron_tempo', tier: 2, visual: 'spark',
    name: 'Iron Tempo', nameUa: 'Залізний ритм',
    desc: 'Requires 2x Iron Nugget – metallic sparks flash', descUa:
'Потрібні 2× Залізна крупинка – металеві іскри',
    coinCost: 0, itemReq: { itemDefId: 'iron_nugget', count: 2 },
    clickBonus: 4, autoBonus: 3, maxLevel: 3,
  },
  // — Tier 3: Rare items —————
  {
    id: 'sapphire_strike', tier: 3, visual: 'sapphire',
    name: 'Sapphire Strike', nameUa: 'Сапфіровий удар',
    desc: 'Requires 1x Sapphire – blue lightning', descUa: 'Потрібен 1×
Сапфір – блискавка',
    coinCost: 0, itemReq: { itemDefId: 'sapphire', count: 1 },
    clickBonus: 20, autoBonus: 0, maxLevel: 2,
  },
  {
    id: 'dragon_engine', tier: 3, visual: 'dragon',
    name: 'Dragon Engine', nameUa: 'Двигун дракона',
    desc: 'Requires 1x Dragon Crystal – fire aura', descUa: 'Потрібен 1×
Кристал дракона – вогняна аура',
    coinCost: 0, itemReq: { itemDefId: 'dragon_crystal', count: 1 },
    clickBonus: 0, autoBonus: 15, maxLevel: 2,
  },
  {
    id: 'sword_slash', tier: 3, visual: 'sword',
    name: 'Sword Slash', nameUa: 'Удар меча',
    desc: 'Requires 1x Iron Sword – sword orbits coin', descUa: 'Потрібен 1×
Залізний меч – меч обертається навколо монети',
    coinCost: 0, itemReq: { itemDefId: 'iron_sword', count: 1 },
    clickBonus: 10, autoBonus: 8, maxLevel: 2,
  },
  {
    id: 'golden_guard', tier: 3, visual: 'shield',
    name: 'Golden Guard', nameUa: 'Золотий щит',
    desc: 'Requires 1x Golden Shield – shields orbit the coin', descUa:
'Потрібен 1× Золотий щит – щити обертаються навколо монети',
    coinCost: 0, itemReq: { itemDefId: 'golden_shield', count: 1 },
    clickBonus: 15, autoBonus: 8, maxLevel: 2,
  },
  // — Tier 4: Epic items —————
  {
    id: 'void_power', tier: 4, visual: 'void',
    name: 'Void Power', nameUa: 'Сила порожнечі',
    desc: 'Requires 1x Void Essence – void swirl', descUa: 'Потрібна 1×
Есенція порожнечі – вихор',
    coinCost: 0, itemReq: { itemDefId: 'void_essence', count: 1 },
    clickBonus: 60, autoBonus: 0, maxLevel: 1,
  },
  {
    id: 'shadow_farm', tier: 4, visual: 'shadow',
    name: 'Shadow Farm', nameUa: 'Тіньова ферма',
    desc: 'Requires 1x Shadow Crown – dark aura', descUa: 'Потрібна 1×
Тіньова корона – темна аура',
    coinCost: 0, itemReq: { itemDefId: 'shadow_crown', count: 1 },
    clickBonus: 0, autoBonus: 40, maxLevel: 1,
  },

```

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		75

```

    },
    {
      id: 'crystal_storm', tier: 4, visual: 'crystal',
      name: 'Crystal Storm', nameUa: 'Кристална буря',
      desc: 'Requires 1x Dragon Crystal – crystals orbit', descUa: 'Потрібен 1x
Кристал дракона – кристали навколо',
      coinCost: 0, itemReq: { itemDefId: 'dragon_crystal', count: 1 },
      clickBonus: 30, autoBonus: 20, maxLevel: 1,
    },
    // — Tier 5: Legendary items —————
    {
      id: 'phoenix_aura', tier: 5, visual: 'phoenix',
      name: 'Phoenix Aura', nameUa: 'Аура фенікса',
      desc: 'Requires 1x Phoenix Feather – flame halo', descUa: 'Потрібне 1x
Перо фенікса – полум'яний ореол',
      coinCost: 0, itemReq: { itemDefId: 'phoenix_feather', count: 1 },
      clickBonus: 100, autoBonus: 25, maxLevel: 1,
    },
    {
      id: 'relic_power', tier: 5, visual: 'relic',
      name: 'Ancient Power', nameUa: 'Стародавня сила',
      desc: 'Requires 1x Ancient Relic – golden rays', descUa: 'Потрібна 1x
Стародавня реліквія – золоті промені',
      coinCost: 0, itemReq: { itemDefId: 'ancient_relic', count: 1 },
      clickBonus: 80, autoBonus: 50, maxLevel: 1,
    },
    {
      id: 'void_storm', tier: 5, visual: 'storm',
      name: 'Void Storm', nameUa: 'Буря порожнечі',
      desc: 'Requires 1x Void Artifact – electric storm erupts', descUa:
'Потрібен 1x Артефакт порожнечі – електрична буря навколо монети',
      coinCost: 0, itemReq: { itemDefId: 'void_artifact', count: 1 },
      clickBonus: 60, autoBonus: 80, maxLevel: 1,
    },
    // — Tier 6: Mythic items —————
    {
      id: 'cosmos_skin', tier: 6, visual: 'cosmos',
      name: 'Cosmic Coin', nameUa: 'Космічна монета',
      desc: 'Requires 1x Cosmos Gem – transforms coin to galaxy!', descUa:
'Потрібен 1x Камінь космосу – перетворює монету на галактику!',
      coinCost: 0, itemReq: { itemDefId: 'cosmos_gem', count: 1 },
      clickBonus: 250, autoBonus: 75, maxLevel: 1,
    },
    {
      id: 'divine_touch', tier: 6, visual: 'divine',
      name: 'Divine Touch', nameUa: 'Божественний дотик',
      desc: 'Requires 1x Divine Relic – divine particles', descUa: 'Потрібна 1x
Божественна реліквія – божественні частинки',
      coinCost: 0, itemReq: { itemDefId: 'divine_relic', count: 1 },
      clickBonus: 300, autoBonus: 100, maxLevel: 1,
    },
  ];

const CONVERT_RATE = 1000;

async function getOrCreateState(userId) {
  const { rows } = await pool.query(
    `INSERT INTO clicker_state (user_id) VALUES ($1)
    ON CONFLICT (user_id) DO UPDATE SET user_id = EXCLUDED.user_id RETURNING
    *`,
    [userId]
  );
};

```

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

```

    return rows[0];
}

function applyAutoCoins(state) {
    const coins = Number(state.coins) || 0;
    if (Number(state.auto_power) <= 0) return { ...state, coins };
    const nowMs = Date.now();
    const lastMs = new Date(state.last_auto_at).getTime();
    const secondsElapsed = Math.min((nowMs - lastMs) / 1000, 3600);
    const earned = Math.floor(Number(state.auto_power) * secondsElapsed);
    return { ...state, coins: coins + earned };
}

async function getSpecificItems(userId, itemDefId, limit) {
    const { rows } = await pool.query(
        `SELECT i.id FROM items i
        LEFT JOIN auction_listings al ON al.item_id = i.id AND al.status =
'ACTIVE'
        WHERE i.user_id = $1 AND i.item_def_id = $2 AND al.id IS NULL LIMIT $3`,
        [userId, itemDefId, limit]
    );
    return rows.map(r => r.id);
}

// GET /api/v1/clicker
router.get('/', requireAuth, async (req, res) => {
    try {
        const raw = await getOrCreateState(req.user.id);
        const state = applyAutoCoins(raw);
        if (state.coins !== Number(raw.coins)) {
            await pool.query(`UPDATE clicker_state SET coins = $1, last_auto_at =
NOW() WHERE user_id = $2`, [state.coins, req.user.id]);
        }
        res.json({
            ok: true,
            coins: Number(state.coins),
            clickPower: Number(state.click_power),
            autoPower: Number(state.auto_power),
            upgrades: state.upgrades,
            convertRate: CONVERT_RATE,
            upgradeList: UPGRADES,
        });
    } catch (err) {
        console.error(err);
        res.status(500).json({ ok: false, error: 'Internal server error' });
    }
});

// POST /api/v1/clicker/click - batched
router.post('/click', requireAuth, async (req, res) => {
    const clicks = Math.min(200, Math.max(1, parseInt(req.body.clicks) || 1));
    try {
        const raw = await getOrCreateState(req.user.id);
        const state = applyAutoCoins(raw);
        const earned = Number(state.click_power) * clicks;
        const newCoins = Number(state.coins) + earned;
        await pool.query(`UPDATE clicker_state SET coins = $1, last_auto_at =
NOW() WHERE user_id = $2`, [newCoins, req.user.id]);
        res.json({ ok: true, coins: Number(newCoins), earned });
    } catch (err) {
        console.error(err);
        res.status(500).json({ ok: false, error: 'Internal server error' });
    }
});

```

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		77

```

    }
  });

// POST /api/v1/clicker/upgrade
router.post('/upgrade', requireAuth, async (req, res) => {
  const { upgradeId } = req.body;
  const upgradeDef = UPGRADES.find(u => u.id === upgradeId);
  if (!upgradeDef) return res.status(400).json({ ok: false, error: 'Unknown upgrade' });

  const client = await pool.connect();
  try {
    await client.query('BEGIN');
    const raw = await getOrCreateState(req.user.id);
    const state = applyAutoCoins(raw);
    const levels = state.upgrades ?? {};
    const currentLevel = Number(levels[upgradeId] ?? 0);

    if (currentLevel >= upgradeDef.maxLevel) {
      await client.query('ROLLBACK');
      return res.status(400).json({ ok: false, error: 'Upgrade maxed out' });
    }

    let newCoins = Number(state.coins);

    if (upgradeDef.itemReq) {
      const { itemDefId, count } = upgradeDef.itemReq;
      const itemIds = await getSpecificItems(req.user.id, itemDefId, count);
      if (itemIds.length < count) {
        await client.query('ROLLBACK');
        const def = ITEM_DEFS[itemDefId];
        return res.status(400).json({ ok: false, error: `Need ${count}x ${def?.name ?? itemDefId}` });
      }
      await client.query(`DELETE FROM items WHERE id = ANY($1) AND user_id = $2`, [itemIds.slice(0, count), req.user.id]);
    } else {
      const cost = upgradeDef.coinCost * (currentLevel + 1);
      if (newCoins < cost) {
        await client.query('ROLLBACK');
        return res.status(400).json({ ok: false, error: 'Not enough coins' });
      }
      newCoins -= cost;
    }

    const newLevels = { ...levels, [upgradeId]: currentLevel + 1 };
    const newClickPower = Number(state.click_power) + upgradeDef.clickBonus;
    const newAutoPower = Number(state.auto_power) + upgradeDef.autoBonus;

    await client.query(
      `UPDATE clicker_state SET coins = $1, click_power = $2, auto_power = $3, upgrades = $4, last_auto_at = NOW() WHERE user_id = $5`,
      [newCoins, newClickPower, newAutoPower, JSON.stringify(newLevels), req.user.id]
    );

    await client.query('COMMIT');
    res.json({ ok: true, coins: Number(newCoins), clickPower: Number(newClickPower), autoPower: Number(newAutoPower), upgrades: newLevels });
  } catch (err) {

```

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		78

```

    await client.query('ROLLBACK');
    console.error(err);
    res.status(500).json({ ok: false, error: 'Internal server error' });
  } finally {
    client.release();
  }
});

// POST /api/v1/clicker/convert
router.post('/convert', requireAuth, async (req, res) => {
  try {
    const raw = await getOrCreateState(req.user.id);
    const state = applyAutoCoins(raw);
    const coins = Number(state.coins);
    if (coins < CONVERT_RATE) return res.status(400).json({ ok: false, error:
`Need at least ${CONVERT_RATE} coins` });
    const batches = Math.floor(coins / CONVERT_RATE);
    const coinsUsed = batches * CONVERT_RATE;
    await pool.query(`UPDATE clicker_state SET coins = coins - $1,
last_auto_at = NOW() WHERE user_id = $2`, [coinsUsed, req.user.id]);
    await addBalance(req.user.id, batches);
    res.json({ ok: true, coinsUsed, kCoinsEarned: batches, coinsLeft: coins -
coinsUsed });
  } catch (err) {
    console.error(err);
    res.status(500).json({ ok: false, error: 'Internal server error' });
  }
});

export default router;

```

Лістинг коду файлу /routes/dice.js

```

import { Router } from 'express';
import { requireAuth, deductBalance, addBalance, recordStats } from
'../middleware/auth.js';
import { pushBigWin } from '../drops.js';
import { DICE_HOUSE_EDGE, BIG_WIN_THRESHOLD } from '../constants/index.js';

const router = Router();

function calcDicePayout(rollOver) {
  const winChance = (100 - rollOver) / 100;
  const multiplier = (1 - DICE_HOUSE_EDGE) / winChance;
  return { winChancePercentage: winChance * 100, payout:
parseFloat(multiplier.toFixed(4)) };
}

// GET /api/v1/games/dice/game/payout?rollOver=X
router.get('/game/payout', (req, res) => {
  const rollOver = parseFloat(req.query.rollOver);
  if (!Number.isFinite(rollOver) || rollOver < 1 || rollOver > 98) {
    return res.status(400).json({ ok: false, error: 'rollOver must be between
1 and 98' });
  }
  res.json(calcDicePayout(rollOver));
});

```

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		79

```
// POST /api/v1/games/dice/game/play
router.post('/game/play', requireAuth, async (req, res) => {
  const bet      = Number(req.body.bet);
  const rollOver = Number(req.body.rollOver);

  if (!Number.isFinite(bet) || bet <= 0) {
    return res.status(400).json({ ok: false, error: 'Invalid bet' });
  }
  if (!Number.isFinite(rollOver) || rollOver < 1 || rollOver > 98) {
    return res.status(400).json({ ok: false, error: 'rollOver must be between
1 and 98' });
  }

  try {
    await deductBalance(req.user.id, bet);
  } catch {
    return res.status(400).json({ ok: false, error: 'Insufficient balance'
});
  }

  const roll    = parseFloat((Math.random() * 99.99).toFixed(2));
  const isWin   = roll > rollOver;
  const { payout: multiplier } = calcDicePayout(rollOver);
  const payout = isWin ? parseFloat((bet * multiplier).toFixed(2)) : 0;

  if (isWin) await addBalance(req.user.id, payout);
  recordStats(req.user.id, 'dice', bet, payout);

  if (isWin && payout >= bet * BIG_WIN_THRESHOLD) {
    pushBigWin({ nick: req.user.nickname, game: 'Dice', amount: payout, mult:
payout / bet });
  }

  res.json({ roll, isWin, payout });
});

export default router;
```

Лістинг коду файлу /routes/drops.js

```
import { Router } from 'express';
import { BIG_WIN_TIERS, getBigWinTier } from '../constants/index.js';

const router = Router();

// Circular buffer of last 50 drops (in-memory)
const recentDrops = [];
const MAX_DROPS = 50;

export function pushDrop(drop) {
  recentDrops.unshift({ ...drop, type: drop.type ?? 'drop' }); // newest
  first
  if (recentDrops.length > MAX_DROPS) recentDrops.pop();
}

/**
 * Push a Big / Mega / Super Win event into the live ticker.
 * Tier is chosen automatically based on the mult ratio.
```

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		80

```

*/
export function pushBigWin({ nick, game, amount, mult }) {
  const tierKey = getBigWinTier(mult);
  const tier = BIG_WIN_TIERS[tierKey];
  pushDrop({
    type: 'bigwin',
    nick,
    icon: tier.icon,
    name: `${tier.label} ${game}`,
    nameUa: `${tier.label} ${game}`,
    rarity: tier.rarity,
    color: tier.color,
    ts: Date.now(),
    amount,
    mult: Math.round(mult),
  });
}

// GET /api/v1/drops/recent
router.get('/recent', (req, res) => {
  res.json(recentDrops.slice(0, 24));
});

export default router;

```

Лістинг коду файлу /routes/items.js

```

import { Router } from 'express';
import pool from '../db.js';
import { requireAuth, addBalance } from '../middleware/auth.js';
import { ITEM_DEFS, getItemDef } from '../itemDefs.js';

const router = Router();

function itemDto(row) {
  const def = ITEM_DEFS[row.item_def_id] ?? {};
  return {
    id: row.id,
    itemDefId: row.item_def_id,
    name: def.name ?? row.item_def_id,
    nameUa: def.nameUa ?? row.item_def_id,
    rarity: def.rarity ?? 'common',
    icon: def.icon ?? '🎲',
    color: def.color ?? '#9ca3af',
    value: def.value ?? 0,
    source: row.source,
    listedAt: row.listed_at ?? null,
    createdAt: row.created_at,
  };
}

// GET /api/v1/items/defs - all item definitions
router.get('/defs', (req, res) => {
  res.json(Object.values(ITEM_DEFS));
});

// GET /api/v1/items/me - user's inventory
router.get('/me', requireAuth, async (req, res) => {

```

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		81

```

try {
  const { rows } = await pool.query(
    `SELECT i.*, al.id as listing_id
    FROM items i
    LEFT JOIN auction_listings al ON al.item_id = i.id AND al.status =
'ACTIVE'
    WHERE i.user_id = $1
    ORDER BY i.created_at DESC`,
    [req.user.id]
  );
  res.json({
    ok: true,
    items: rows.map(r => ({ ...itemDto(r), listingId: r.listing_id ?? null
})),
  });
} catch (err) {
  console.error(err);
  res.status(500).json({ ok: false, error: 'Internal server error' });
}
});

// POST /api/v1/items/:itemId/sell-vendor – sell item to vendor at 50% base
value
router.post('/:itemId/sell-vendor', requireAuth, async (req, res) => {
  try {
    // Check item exists and belongs to this user
    const { rows } = await pool.query(
      `SELECT i.* FROM items i
      LEFT JOIN auction_listings al ON al.item_id = i.id AND al.status =
'ACTIVE'
      WHERE i.id = $1 AND i.user_id = $2`,
      [req.params.itemId, req.user.id]
    );
    if (!rows[0]) return res.status(404).json({ ok: false, error: 'Item not
found or not yours' });

    const item = rows[0];
    const def = getItemDef(item.item_def_id);
    const baseValue = def?.value ?? 0;
    const sellPrice = Math.floor(baseValue * 0.5);

    if (sellPrice <= 0) return res.status(400).json({ ok: false, error: 'Item
has no sell value' });

    // Delete item and credit balance atomically
    await pool.query(`DELETE FROM items WHERE id = $1`, [item.id]);
    const newBalance = await addBalance(req.user.id, sellPrice);

    res.json({ ok: true, gained: sellPrice, balance: newBalance });
  } catch (err) {
    console.error(err);
    res.status(500).json({ ok: false, error: 'Internal server error' });
  }
});

export default router;

```

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		82

Лістинг коду файлу /routes/mines.js

```
import { Router } from 'express';
import pool from '../db.js';
import { requireAuth, addBalance } from '../middleware/auth.js';
import { ITEM_DEFS, getItemDef } from '../itemDefs.js';

const router = Router();

function itemDto(row) {
  const def = ITEM_DEFS[row.item_def_id] ?? {};
  return {
    id: row.id,
    itemDefId: row.item_def_id,
    name: def.name ?? row.item_def_id,
    nameUa: def.nameUa ?? row.item_def_id,
    rarity: def.rarity ?? 'common',
    icon: def.icon ?? '📦',
    color: def.color ?? '#9ca3af',
    value: def.value ?? 0,
    source: row.source,
    listedAt: row.listed_at ?? null,
    createdAt: row.created_at,
  };
}

// GET /api/v1/items/defs - all item definitions
router.get('/defs', (req, res) => {
  res.json(Object.values(ITEM_DEFS));
});

// GET /api/v1/items/me - user's inventory
router.get('/me', requireAuth, async (req, res) => {
  try {
    const { rows } = await pool.query(
      `SELECT i.*, al.id as listing_id
      FROM items i
      LEFT JOIN auction_listings al ON al.item_id = i.id AND al.status =
'ACTIVE'
      WHERE i.user_id = $1
      ORDER BY i.created_at DESC`,
      [req.user.id]
    );
    res.json({
      ok: true,
      items: rows.map(r => ({ ...itemDto(r), listingId: r.listing_id ?? null })),
    });
  } catch (err) {
    console.error(err);
    res.status(500).json({ ok: false, error: 'Internal server error' });
  }
});

// POST /api/v1/items/:itemId/sell-vendor - sell item to vendor at 50% base value
router.post('/:itemId/sell-vendor', requireAuth, async (req, res) => {
  try {
    // Check item exists and belongs to this user
  }
});
```

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		83

```

const { rows } = await pool.query(
  `SELECT i.* FROM items i
    LEFT JOIN auction_listings al ON al.item_id = i.id AND al.status =
'ACTIVE'
    WHERE i.id = $1 AND i.user_id = $2`,
  [req.params.itemId, req.user.id]
);
if (!rows[0]) return res.status(404).json({ ok: false, error: 'Item not
found or not yours' });

const item = rows[0];
const def = getItemDef(item.item_def_id);
const baseValue = def?.value ?? 0;
const sellPrice = Math.floor(baseValue * 0.5);

if (sellPrice <= 0) return res.status(400).json({ ok: false, error: 'Item
has no sell value' });

// Delete item and credit balance atomically
await pool.query(`DELETE FROM items WHERE id = $1`, [item.id]);
const newBalance = await addBalance(req.user.id, sellPrice);

res.json({ ok: true, gained: sellPrice, balance: newBalance });
} catch (err) {
  console.error(err);
  res.status(500).json({ ok: false, error: 'Internal server error' });
}
});

export default router;

```

Лістинг коду файлу /routes/plinko.js

```

import { Router } from 'express';
import { requireAuth, deductBalance, addBalance, recordStats } from
'../middleware/auth.js';
import { pushBigWin } from '../drops.js';
import { BIG_WIN_THRESHOLD } from '../constants/index.js';

const router = Router();

// Multiplier tables – identical to frontend baseTables fallback
const TABLES = {
  'LOW:8': [3, 1.5, 1.1, 1.0, 0.7, 1.0, 1.1, 1.5, 3],
  'MEDIUM:8': [8, 2.0, 1.3, 0.7, 0.3, 0.7, 1.3, 2.0, 8],
  'HIGH:8': [29, 4.0, 1.5, 0.5, 0.2, 0.5, 1.5, 4.0, 29],

  'LOW:12': [5, 2.0, 1.6, 1.3, 1.1, 1.0, 0.8, 1.0, 1.1, 1.3, 1.6, 2.0, 5],
  'MEDIUM:12': [29, 8.0, 3.0, 1.6, 1.1, 0.8, 0.5, 0.8, 1.1, 1.6, 3.0, 8.0,
29],
  'HIGH:12': [120, 26, 8.0, 2.0, 1.0, 0.6, 0.2, 0.6, 1.0, 2.0, 8.0, 26,
120],

  'LOW:16': [10, 3, 2, 1.6, 1.3, 1.1, 1.0, 0.8, 0.7, 0.8, 1.0, 1.1, 1.3,
1.6, 2, 3, 10],
  'MEDIUM:16': [110, 41, 10, 5, 3, 1.5, 1, 0.5, 0.3, 0.5, 1, 1.5, 3, 5, 10,
41, 110],
  'HIGH:16': [1000, 110, 41, 10, 5, 2, 1, 0.5, 0.2, 0.5, 1, 2, 5, 10, 41,
110, 1000],

```

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		84

```

};

function popcount(x) {
  x = x >>> 0;
  let c = 0;
  while (x) { x &= x - 1; c++; }
  return c;
}

function getTable(rows, difficulty) {
  return TABLES[`${String(difficulty).toUpperCase()}:${rows}`] ?? null;
}

// GET /api/v1/games/plinko/game/multipliers?rows=X&difficulty=Y
router.get('/game/multipliers', (req, res) => {
  const rows = parseInt(req.query.rows);
  const difficulty = String(req.query.difficulty || '').toUpperCase();
  const table = getTable(rows, difficulty);
  if (!table) return res.status(400).json({ ok: false, error: 'Invalid rows or difficulty' });
  res.json(table);
});

// POST /api/v1/games/plinko/game/play
router.post('/game/play', requireAuth, async (req, res) => {
  const { bet, balls, rows, difficulty } = req.body;

  if (!bet || bet <= 0) return res.status(400).json({ ok: false, error: 'Invalid bet' });
  const n = Math.max(1, Math.min(100, parseInt(balls) || 1));
  const table = getTable(rows, difficulty);
  if (!table) return res.status(400).json({ ok: false, error: 'Invalid rows or difficulty' });

  const totalBet = parseFloat((bet * n).toFixed(2));
  try {
    await deductBalance(req.user.id, totalBet);
  } catch {
    return res.status(400).json({ ok: false, error: 'Insufficient balance' });
  }

  const traces = [];
  let totalWin = 0;

  for (let i = 0; i < n; i++) {
    let mask = 0;
    for (let r = 0; r < rows; r++) {
      if (Math.random() < 0.5) mask |= (1 << r);
    }
    const landing = popcount(mask);
    const multiplier = table[landing] ?? 0;
    const win = parseFloat((bet * multiplier).toFixed(2));
    traces.push({ win, mask });
    totalWin = parseFloat((totalWin + win).toFixed(2));
  }

  if (totalWin > 0) await addBalance(req.user.id, totalWin);
  recordStats(req.user.id, 'plinko', totalBet, totalWin);

  if (totalBet > 0 && totalWin >= totalBet * BIG_WIN_THRESHOLD) {
    pushBigWin({ nick: req.user.nickname, game: 'Plinko', amount: totalWin,

```

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		85

```

    mult: totalWin / totalBet });
  }

  res.json({ total: totalWin, traces });
});

export default router;

```

Лістинг коду файлу /routes/roulette.js

```

import { Router } from 'express';
import { requireAuth, deductBalance, addBalance, recordStats } from
'../middleware/auth.js';
import { pushBigWin } from '../drops.js';
import { BIG_WIN_THRESHOLD } from '../constants/index.js';

const router = Router();

const RED_NUMBERS = new
Set([1, 3, 5, 7, 9, 12, 14, 16, 18, 19, 21, 23, 25, 27, 30, 32, 34, 36]);
const COLUMN_1 = new Set([1, 4, 7, 10, 13, 16, 19, 22, 25, 28, 31, 34]);
const COLUMN_2 = new Set([2, 5, 8, 11, 14, 17, 20, 23, 26, 29, 32, 35]);
const COLUMN_3 = new Set([3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36]);

const VALID_KEYS = new Set([
  'red', 'black', 'even', 'odd', 'low', 'high',
  'range:1', 'range:2', 'range:3',
  'row:1', 'row:2', 'row:3',
  ...Array.from({ length: 37 }, (_, i) => `n:${i}`),
]);

function resolveBet(key, number) {
  if (number === 0) return key === 'n:0' ? 36 : 0;
  if (key.startsWith('n:')) {
    const n = parseInt(key.slice(2));
    return (Number.isInteger(n) && n >= 0 && n <= 36 && number === n) ? 36 :
0;
  }
  switch (key) {
    case 'red': return RED_NUMBERS.has(number) ? 2 : 0;
    case 'black': return !RED_NUMBERS.has(number) ? 2 : 0;
    case 'even': return number % 2 === 0 ? 2 : 0;
    case 'odd': return number % 2 !== 0 ? 2 : 0;
    case 'low': return number >= 1 && number <= 18 ? 2 : 0;
    case 'high': return number >= 19 && number <= 36 ? 2 : 0;
    case 'range:1': return number >= 1 && number <= 12 ? 3 : 0;
    case 'range:2': return number >= 13 && number <= 24 ? 3 : 0;
    case 'range:3': return number >= 25 && number <= 36 ? 3 : 0;
    case 'row:1': return COLUMN_1.has(number) ? 3 : 0;
    case 'row:2': return COLUMN_2.has(number) ? 3 : 0;
    case 'row:3': return COLUMN_3.has(number) ? 3 : 0;
    default: return 0;
  }
}

// POST /api/v1/games/roulette/game/play
router.post('/game/play', requireAuth, async (req, res) => {
  const { bets } = req.body;
  if (!Array.isArray(bets) || bets.length === 0) {

```

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		86

```

    return res.status(400).json({ ok: false, error: 'bets must be a non-empty
array' });
  }
  if (bets.length > 100) {
    return res.status(400).json({ ok: false, error: 'Too many bets' });
  }

  for (const b of bets) {
    const amt = Number(b.amount);
    if (!Number.isFinite(amt) || amt <= 0) {
      return res.status(400).json({ ok: false, error: 'Each bet amount must
be a positive number' });
    }
    if (!VALID_KEYS.has(b.key)) {
      return res.status(400).json({ ok: false, error: `Invalid bet key:
${b.key}` });
    }
  }

  const totalBet = bets.reduce((s, b) => s + Number(b.amount), 0);
  if (!Number.isFinite(totalBet) || totalBet <= 0) {
    return res.status(400).json({ ok: false, error: 'Invalid total bet' });
  }

  try {
    await deductBalance(req.user.id, totalBet);
  } catch {
    return res.status(400).json({ ok: false, error: 'Insufficient balance'
});
  }

  const number = Math.floor(Math.random() * 37);
  let totalReturn = 0;
  for (const { key, amount } of bets) {
    totalReturn += Number(amount) * resolveBet(key, number);
  }

  if (totalReturn > 0) await addBalance(req.user.id, totalReturn);
  recordStats(req.user.id, 'roulette', totalBet, totalReturn);

  if (totalBet > 0 && totalReturn >= totalBet * BIG_WIN_THRESHOLD) {
    pushBigWin({ nick: req.user.nickname, game: 'Roulette', amount:
totalReturn, mult: totalReturn / totalBet });
  }

  const netAmount = parseFloat((totalReturn - totalBet).toFixed(2));
  res.json({ number, amount: netAmount });
});
export default router;

```

		Бігун Р.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		87