Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавра

зі спеціальності 121 «Інженерія програмного забезпечення»

На тему: *Розробка навчальної комп'ютерної гри-головоломки «Математичні кросворди»*

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ-21-2
_____ *Окороков Г. В.*

Керівник кваліфікаційної роботи _____ */ Стрюк А. М. /*

Завідувач кафедри _____ */ Стрюк А. М. /*

Кривий Ріг
2025

Криворізький національний університет
Факультет: *Інформаційних технологій*
Кафедра: *Моделювання та програмного забезпечення*
Освітньо-кваліфікаційний рівень: *бакалавр*
Спеціальність: *121 "Інженерія програмного забезпечення"*

ЗАТВЕРДЖУЮ
Зав. кафедри
_____*Стрюк А. М.*
«___»_____ 2025__ р.

ЗАВДАННЯ
на кваліфікаційну роботу
студенту групи ІПЗ-21-2 Окорокову Герману Віталійовичу

1. Тема: *Розробка навчальної комп'ютерної гри-головоломки «Математичні кросворди»* затверджено наказом по КНУ №119 від «10» квітня 2025 р.
2. Термін подання студентом закінченого проекту «20» червня 2025 р.
3. Вихідні дані по роботі: *Пояснювальна записка: 114 сторінок, 26 рисунків, 1 додаток, 11 використаних у роботі джерел*
4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): актуальність розробки навчальної гри. Проектування програмного забезпечення гри. Розробка та тестування програмного забезпечення.
5. Перелік графічного демонстраційного матеріалу: Приклади існуючих програм. Діаграма потоків даних. Діаграми використання. Блок-схеми основних алгоритмів. Структура бази даних. Приклади роботи розробленої програми.

Календарний план:

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	*Формулювання мети та задач роботи*	*13.02.2025-20.02.2025*
2	*Аналіз інформаційних джерел*	*21.02.2025-09.03.2025*
3	*Визначення вимог до програмного забезпечення*	*10.03.2025-18.03.2025*
4	*Розробка функціональної схеми*	*19.03.2025-23.03.2025*
5	*Розробка алгоритмів*	*24.03.2025-31.03.2025*
6	*Розробка бази даних*	*01.04.2025-14.04.2025*
7	*Розробка програмного забезпечення*	*15.04.2025-13.05.2025*
8	*Тестування програмного забезпечення*	*14.05.2025-20.05.2025*
9	*Оформлення пояснювальної записки*	*21.05.2025-12.06.2025*
10	*Розробка демонстраційних матеріалів*	*13.06.2025-17.06.2025*

Дата видачі завдання: «_____»_____ 2025 р.
Студент _____ Окороков Г. В.
Керівник роботи _____ Стрюк А. М.

РЕФЕРАТ

МАТЕМАТИЧНІ ІГРИ, МАТЕМАТИЧНІ ГОЛОВОЛОМКИ, КОМП’ЮТЕРНА ГРА, ВЕБ-ЗАСТОСУНОК, JAVASCRIPT, REACT

Кваліфікаційна робота містить 114 сторінок, 26 рисунків, 11 джерел, 1 додаток.

Метою кваліфікаційної роботи є створення навчальної комп’ютерної гри-головоломки «Математичні кросворди».

В роботі проаналізовано математичні кросворди як засіб навчання, вивчена популярність існуючих математичних ігор. Доведено актуальність розробки навчальної комп’ютерної гри-головоломки «Математичні кросворди». Визначенно вимоги до навчальної комп’ютерної гри-головоломки, спроєктовані основні алгоритми, змодельовано діаграму використання. Навчальну гру реалізовано за допомогою мови JavaScript та бібліотеки React.

Створена гра може використовуватись як засіб навчання та розвитку математичних навичок в грі.

ABSTRACT

MATHEMATICAL GAMES, MATHEMATICAL PUZZLES, COMPUTER GAME, WEB APPLICATION, JAVASCRIPT, REACT

The qualification work comprises 114 pages, 26 figures, 11 references, and 1 appendix.

The goal of the qualification work is to develop an educational computer puzzle game titled 'Mathematical Crosswords.'

The study analyzes mathematical crosswords as a learning tool and examines the popularity of existing mathematical games. The relevance of developing the educational computer puzzle game 'Mathematical Crosswords' is proven. Requirements for the educational computer puzzle game are defined, key algorithms are designed, and a use-case diagram is modeled. The educational game is implemented using JavaScript and the React library.

The developed game can be used as a tool for learning and enhancing mathematical skills through gameplay.

ЗМІСТ

Вступ 7

1 АКТУАЛЬНІСТЬ РОЗРОБКИ НАВЧАЛЬНОЇ КОМП'ЮТЕРНОЇ ГРИ-ГОЛОВОЛОМКИ «МАТЕМАТИЧНІ КРОСВОРДИ» 9

1.1 Математичні кросворди як засіб навчання 9

1.2 Аналіз існуючих математичних ігор 11

1.3 Визначення вимог до навчальної комп'ютерної гри-головоломки «Математичні кросворди» 23

2 ПРОЕКТУВАННЯ НАВЧАЛЬНОЇ КОМП'ЮТЕРНОЇ ГРИ-ГОЛОВОЛОМКИ «МАТЕМАТИЧНІ КРОСВОРДИ» 35

2.1 Моделювання гри «Математичні кросворди» 35

2.2 Проєктування основних алгоритмів 39

3 РЕАЛІЗАЦІЯ НАВЧАЛЬНОЇ КОМП'ЮТЕРНОЇ ГРИ-ГОЛОВОЛОМКИ «МАТЕМАТИЧНІ КРОСВОРДИ» 49

3.1 Добір засобів розробки навчальної комп'ютерної гри-головоломки «Математичні кросворди» 49

3.2 Огляд результатів розробки навчальної комп'ютерної гри-головоломки «Математичні кросворди» 51

ВИСНОВКИ 62

Перелік посилань 63

Додаток А Текст програми 64

ВСТУП

Математичні кросворди – це чудовий інструмент для навчання та розваги, який допомагає покращити математичні здібності, розвиток логічного мислення, уваги та інтелектуальних навичок у цікавій, захопливій та інтерактивній формі. Вони поєднують у собі елементи гри та навчання, що робить процес вирішення задач максимально продуктивним та приємним.

Кросворди можуть містити як прості арифметичні приклади, так і складніші математичні рівняння, що стимулює учнів або любителів математики до глибшого розуміння числових закономірностей, операцій та структур. Крім того, вони сприяють розвитку концентрації, терпіння та навичок стратегічного мислення. Використовуються як в освітньому процесі для закріплення знань, так і в домашньому навчанні, дозволяючи дітям чи дорослим з користю проводити вільний час, одночасно вдосконалюючи свої математичні здібності.

Розв’язання математичних кросвордів є чудовим тренуванням для розуму, яке допомагає покращити логічне мислення, увагу до деталей та математичні навички. Цей інтерактивний процес завжди викликає неабияку зацікавленість серед тих, хто прагне поєднати розвагу з навчанням. Проте створення таких завдань, особливо з врахуванням правильного балансу між складністю та цікавістю, часто може бути справжнім викликом. Рутинність процесу постановки умов задач, підбору чисел та формул або перевірки коректності розв’язків може зменшити мотивацію навіть у найзавзятіших ентузіастів.

Саме тому виникає ідея автоматизації цього процесу, яка відкриває нові можливості для створення інтелектуально насичених ігор. Застосування алгоритмів дозволить генерувати математичні кросворди зі зростаючою складністю, відповідно до рівня підготовки гравця. Під час такої гри учасникам пропонується розв’язувати завдання, які стають дедалі більш

складними, що підтримує їхній інтерес і сприяє поступовому розвитку навичок.

Метою нашої роботи є створення навчальної комп'ютерної гри-головоломки «Математичні кросворди».

Ця програма матиме і навчальне, і розважальне значення. Віна допоможе гравцям тренувати мислення та кращу концентрацію, а ще робить математику простішою і веселішою. У майбутньому можна створити велику онлайн-платформу, де люди зможуть показувати свої досягнення, змагатися між собою або разом розв'язувати математичні кросворди.

1 АКТУАЛЬНІСТЬ РОЗРОБКИ НАВЧАЛЬНОЇ КОМП'ЮТЕРНОЇ ГРИ-ГОЛОВОЛОМКИ «МАТЕМАТИЧНІ КРОСВОРДИ»

1.1 Математичні кросворди як засіб навчання

Крім звичайних кросвордів зі словами, існують також захопливі математичні кросворди. Це варіації традиційних кросвордів, де замість букв у клітинах використовуються числа та математичні операції. Розв'язування таких головоломок вимагає не лише знання арифметики, але й логічного мислення та стратегічного планування.

Основними елементами математичних кросвордів є сітка, числа та арифметичні операції.

Як і у звичайних кросвордах, математичні кросворди складаються з сітки, що містить зафарбовані та незафарбовані клітини. У незафарбовані клітини потрібно вписати числа. Ці числа можуть бути цілими, раціональними (дроби), або навіть ірраціональними (хоча це рідше зустрічається в базових версіях). Між числами в горизонтальних або вертикальних рядах розташовуються символи арифметичних операцій:

+ (додавання)

− (віднімання)

× (множення)

÷ (ділення)

= (рівність) - часто використовується для відображення результату обчислення.

Підказки/Вирази: Замість текстових підказок, як у звичайних кросвордах, тут надаються математичні вирази або рівняння, які потрібно заповнити, щоб вони були вірними.

Можуть бути додаткові обмеження, такі як:

– Використання чисел лише з певного діапазону (наприклад, від 1 до 9).

– Використання кожної цифри лише один раз у певному рядку або стовпці (подібно до судоку).

– Наявність унікального розв'язку.

Є декілька різновидів математичних кросвордів:

– Класичні математичні кросворди: У цих кросвордах кожен рядок і стовпець представляє собою математичний вираз, який має бути вірним. Зазвичай, заповнюються числа, а операції вже задані або також потребують вибору з певного набору.

– КенКен (KenKen): Це популярний вид математичних кросвордів, де сітка розбита на "клітки" (групи клітин, обведені товстими лініями). У кожній клітці вказано цільове число та арифметична операція. Наприклад, клітка з "10x" означає, що числа в цій клітці, помножені одне на одного, дадуть 10. Додаткове правило: жодна цифра не може повторюватися в одному рядку або стовпці.

– Кросворди з рівняннями: Деякі кросворди можуть містити більш складні алгебраїчні рівняння, де потрібно знайти невідомі змінні.

– Кросворди з числовими послідовностями: У цих кросвордах потрібно заповнити пропущені числа в числових послідовностях, які можуть бути арифметичними, геометричними або мати іншу закономірність.

Розв'язування математичних кросвордів має певні переваги:

– Розвиток математичних навичок: Покращують навички додавання, віднімання, множення, ділення, а також роботи з дробами та відсотками.

– Логічне мислення: Вимагають аналізу, висування гіпотез, перевірки та стратегічного планування для знаходження правильного розв'язку.

– Концентрація та увага: Допомагають розвинути зосередженість та терпіння.

– Вирішення проблем: Вчать знаходити різні підходи до вирішення однієї задачі.

– Розвиток критичного мислення: Спонукають до обдумування та перевірки своїх дій.

В класичному варіанті математичні кросворди створюються вручну. Для цього потрібно:

– Визначити розмір сітки: Вибрати кількість рядків та стовпців.

– Розмістити операції та знаки рівності: Визначити, де будуть розташовані арифметичні операції та результати.

– Створити вирази/рівняння: Заповнити сітку числами та операціями таким чином, щоб утворилися вірні математичні вирази.

– Вилучити деякі числа/операції: Сховати частину інформації, щоб головоломка мала унікальний розв'язок. Важливо переконатися, що розв'язок є однозначним.

Надати підказки: Чітко сформулювати, що саме потрібно знайти (наприклад, "заповніть порожні клітини числами від 1 до 9").

Проте з часом математичні кросворди еволюціонували і стали різновидом інтерактивних навчальних комп’ютерних ігор, що поєднують розв’язання логічних задач із сучасними технологіями. Такі ігри дозволяють дітям і дорослим вдосконалювати математичні навички у цікавій ігровій формі, сприяючи розвитку логіки, уважності та креативного мислення. Завдяки адаптивним алгоритмам рівень складності завдань у цих іграх поступово зростає, забезпечуючи комфортний і ефективний навчальний процес для користувачів різного віку. Окрім того, сучасні математичні комп’ютерні ігри нерідко мають інтеграцію з освітніми платформами, що дозволяє використовувати їх як частину навчальних програм у школах або для самостійного опанування знань.

1.2 Аналіз існуючих математичних ігор

Математичні комп'ютерні ігри пройшли шлях від простих вправ для тренування навичок до складних головоломок, які допомагають розвивати мислення та вирішувати задачі. Наприклад, ігри типу Crossmath, Kakuro або KenKen особливо популярні за свою цікаву механіку та користь для навчання. Вони допомагають краще освоїти арифметику, покращити логіку та вміння розв'язувати математичні проблеми. Але є й недоліки: у багатьох безкоштовних додатках надто багато реклами, що може псувати враження від

гри. Загалом, такі ігри підходять для навчання людей різного віку, адже вони роблять вивчення математики захопливим і корисним. Якщо обирати їх з розумом, вони можуть добре вплинути на розвиток математичних та логічних навичок.

Комп'ютерні ігри з математики бувають різними: вони підходять як малюкам, так і дорослим, і можуть навчати простих рахунків або складних задач. Раніше такі ігри були простішими, як, наприклад, Number Munchers (1986), Basic Math (1977) і Math Blaster! (1983), де гравці тренували арифметику в стилі аркадних ігор. З часом ці ігри стали цікавішими та більш складними. Нові ігри, такі як Prodigy Math Game (2011), Euclidea, HyperRogue (2015), 2048 (2014), DragonBox: Numbers (2015) і DragonBox: Algebra 12+ (2013), пропонують більше логіки, складні завдання та індивідуальний підхід до навчання.

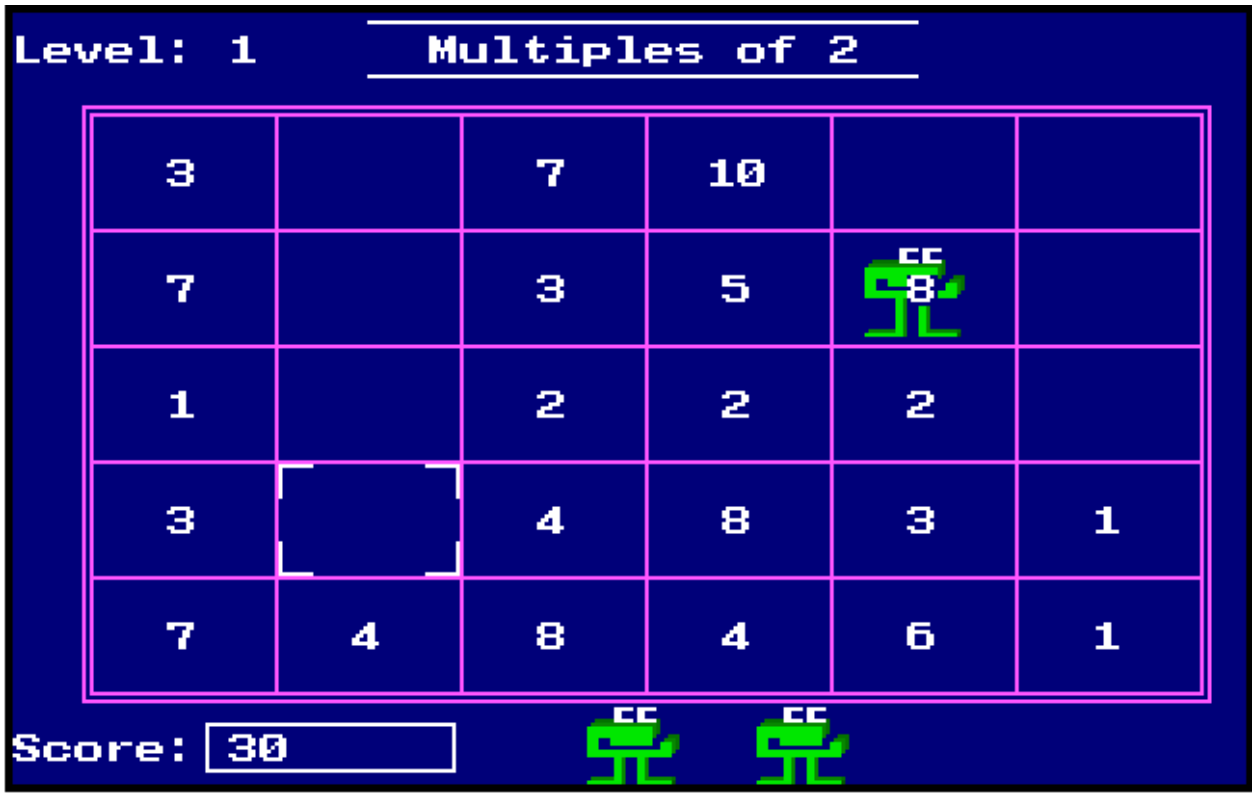

Рисунок 1.1 – Number Munchers (1986)

Рисунок 1.2 – Math Blaster! (1983)

Рисунок 1.3 – Prodigy Math Game

Рисунок 1.4 – 6. DragonBox: Numbers

Ці ігри охоплюють широкий спектр математичних концепцій. Приклади включають базову арифметику (Math Land: Kids Addition Games, Fraction Munchers), геометрію (Euclidea), алгебру (DragonBox: Algebra 12+), показники степеня (2048) та просунуту логіку (Logical Journey of the Zoombinis, Miegakure, Planarity). Це вказує на те, що цифрові ігри розробляються для підтримки навчання в різних математичних областях та рівнях складності.

Зараз все більше звертають увагу на ігри, які допомагають розвивати логічне мислення та вміння вирішувати складні задачі, а не просто рахувати. Це показує, що у навчанні математики почали більше приділяти увагу складнішому мисленню. Популярними стали такі ігри-головоломки, як Crossmath - Math Puzzle Games, Sudoku, HyperRogue, Number Munchers, Conway's Game of Life, Hexologic, Bomb Club Deluxe, Minesweeper та 2048. У них потрібно думати над числами, шукати закономірності та планувати стратегію, тому вони добре тренують мозок. Ці ігри доступні на різних пристроях — комп'ютерах, телефонах (Android, iOS), а також в інтернеті, тому їх легко використовувати і в школі, і вдома для навчання.

Деякі ігри, такі як Prodigy Math Game, поєднують математичні завдання з цікавими сюжетами в жанрі фантастики чи пригод. У таких іграх, щоб просунутися далі або отримати винагороду, потрібно розв'язувати математичні задачі. Це називається гейміфікацією — коли навчання стає схожим на гру.

Інші ігри, як 2048, Sudoku, Kakuro чи KenKen, побудовані саме на математичних головоломках. У них весь процес гри — це вже розв’язання задач. Гравці отримують задоволення просто від того, що вирішують різноманітні математичні або логічні проблеми.

Ці два типи підходів до навчання математики мають значення. Хоча обидва можуть бути корисними, ігри-головоломки часто дають більш глибоке розуміння математики. Вони показують красу та структуру чисел і ідей, що може більше мотивувати людей вивчати математику, не лише заради винагороди, а через інтерес до самої науки.

Математичні кросворди – це цікава версія звичайних кросвордів, але замість слів потрібно розв'язувати завдання з числами та рівняннями. У них важливо не лише добре знати математику, але й вміти логічно думати та планувати. Замість того, щоб здогадуватися про слова, тут шукають правильні числа, щоб заповнити сітку.

Crossmath - Math Puzzle Games

За своєю суттю, Crossmath пропонує гравцям розв'язувати "крос-математичні головоломки та кросворди" шляхом стратегічного заповнення чисел та математичних операторів (додавання, віднімання, множення та ділення) для завершення заданих рівнянь. Вона розроблена як "захоплююча гра-головоломка з числами", яка ретельно перевіряє математичні навички, одночасно відточуючи логіку та критичне мислення. Гра пропонує прогресивний виклик з кількома рівнями складності: Легкий, Середній, Важкий та Експерт. Для підвищення залученості та можливості повторного відтворення вона включає різні режими, такі як щоденний математичний

кросворд, нескінченний режим (де помилки не перевіряються до остаточного подання відповідей, що винагороджує вищими балами за безпомилкове завершення), тематичні числові події та змагання "Гонка зірок математичних головоломок". Користувацький досвід підкріплюється практичними функціями, такими як необмежена кількість скасувань, детальна статистика для відстеження прогресу, великі шрифти та темний режим для покращеного комфорту перегляду, таблиця лідерів для конкурентних гравців та корисні посилення (наприклад, підказки, розширені нотатки) для допомоги у складних головоломках.

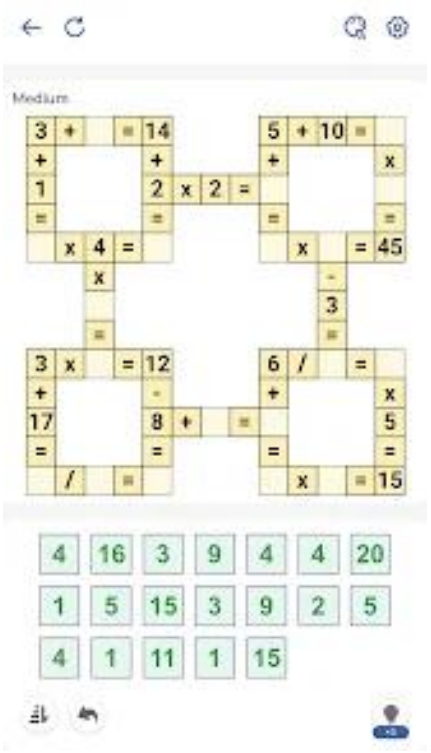

Рисунок 1.5 – Crossmath - Math Puzzle Games

Crossmath позиціонується як інструмент, що допомагає гравцям "швидко освоїти логічні навички та стати майстром математики". Вона рекламується як "веселий та корисний спосіб тренувати свій мозок". Фізична настільна версія, MindWare CrossMath, прямо заявляє про свою роль у розвитку навичок ментальної математики, сприянні стратегічному мисленню та підвищенні впевненості у молодих учнів, що робить її ідеальною для дітей віком від 7 років. Гра широко орієнтована на "Всіх" та конкретно на "Дітей" (для додатку,

для настільної гри віком 7-12 років). Її масштабовані рівні складності забезпечують придатність для широкої демографічної групи, від початківців до просунутих любителів математики. Crossmath широко доступна, включаючи ПК через Google Play Store , та на мобільних платформах (Android, iOS). Примітно, що вона також існує як відчутна, багатокористувацька фізична настільна гра (MindWare CrossMath).

Мобільний додаток має понад 10 мільйонів завантажень та загалом високі оцінки (4,8 зірки в Google Play ; 4,5 зірки в Apple App Store). Однак, відгуки користувачів часто вказують на значні проблеми з надмірною та нав'язливою рекламою, яка часто відтворюється навіть коли телефон вимкнений або після того, як користувачі вирішили "продовжити" без бонусних винагород. Деякі користувачі також повідомляють про перегрів пристрою. Настільна гра MindWare отримала 5 із 5 зірок від єдиного відгуку на Walmart.com.

Kakuro (Крос-Суми)

Kakuro, також відома як "Крос-Суми", є "кросвордом, що використовує числа". Мета полягає в тому, щоб заповнити сітку одноцифровими числами (від 1 до 9) таким чином, щоб сума чисел у кожному горизонтальному та вертикальному "слові" дорівнювала числовій підказці, наданій на початку цього рядка або стовпця. Важливе правило полягає в тому, що жодна цифра не може повторюватися в одному "слові". Підказки зазвичай подаються у трикутній формі на початку кожного числового "слова".

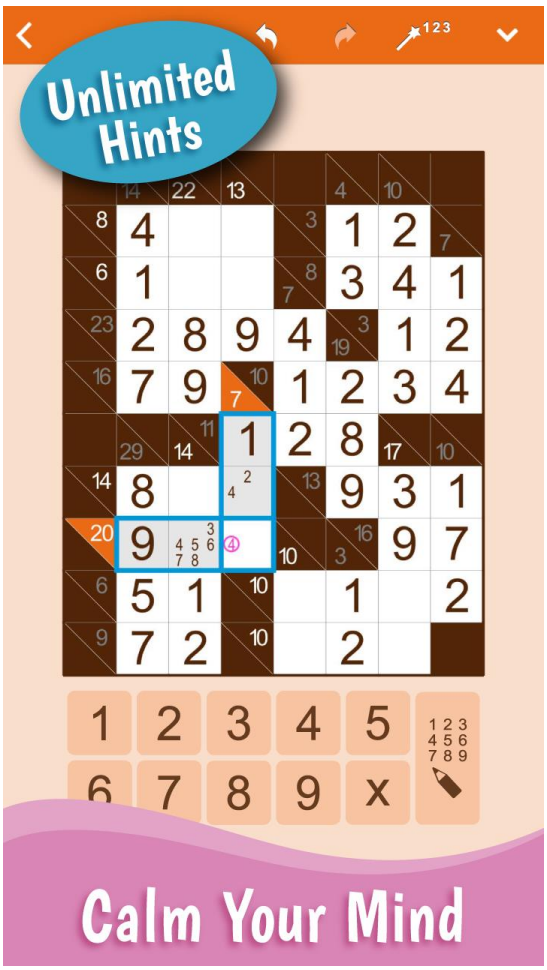

Рисунок 1.6 – Kakuro: Number Crossword

Основною математичною концепцією, що лежить в основі Kakuro, є додавання. Крім простих обчислень, гра активно залучає логічну дедукцію, комбінаторне мислення та критичне мислення для ідентифікації унікальних комбінацій чисел та виключення можливостей на основі правила "без повторень".

Kakuro вважається "однією з найідеальніших ігор для використання на уроках математики". Вона особливо ефективна для покращення швидкості та точності базових навичок додавання та підвищення навичок логічного мислення. Гра допомагає дітям зрозуміти ігрову механіку та ефективно застосовувати додавання. Вона може слугувати цінною додатковою діяльністю або для додаткових завдань в освітніх установах. Наявність простих аркушів головоломок відзначається як важлива для формування впевненості та заохочення змагального духу у молодих учнів. Зрештою, вона

заохочує незалежне мислення та наполегливість. Kakuro описується як "більший хіт у Японії, ніж Судоку!", що підкреслює її значну світову привабливість. Популярні версії додатків включають "Kakuro U" (4,5 зірки, 283 оцінки на iPad) та "Kakuro HD" на Amazon, яка отримує позитивні відгуки за свій складний, але доступний характер, приємний дизайн та ефективність у відточуванні базових математичних навичок та логіки. Користувачі часто порівнюють її якості, що стимулюють мозок, із Судоку.

KenKen

Винайдена у 2004 році японським учителем математики Тецуя Міямото, KenKen є сітковою головоломкою, схожою на Судоку, але з доданим арифметичним компонентом. Мета полягає в тому, щоб заповнити сітку (розміром від 3x3 до 9x9) цифрами (наприклад, 1-4 для сітки 4x4, 1-6 для сітки 6x6) так, щоб жодна цифра не повторювалася в жодному рядку чи стовпці. Що відрізняє KenKen, так це її "клітки" — жирно обведені групи клітинок — кожна з яких містить цільове число та вказану математичну операцію (додавання, віднімання, множення або ділення), яку числа в цій клітці повинні задовольняти. На відміну від Судоку, цифри можуть повторюватися в межах клітки, за умови, що вони не знаходяться в тому ж рядку або стовпці.

KenKen дуже ефективна для відточування елементарних арифметичних навичок та логічного мислення. Вона спонукає гравців застосовувати базові математичні операції, використовуючи логіку, критичне мислення та стратегічну дедукцію. Гра розвиває "логічну дедукцію, пошук життєво важливої інформації та зміну стратегій". KenKen широко визнана за свою освітню корисність: понад 30 000 вчителів у Сполучених Штатах використовують її для навчання математичних навичок, методів вирішення проблем, логіки та критичного мислення. Вона сприяє абстрактному мисленню та заохочує учнів розвивати міцні навички вирішення проблем. Гру хвалять за розвиток незалежного мислення, міркування, концентрації та наполегливості. Вона вважається "відмінною практикою арифметики".

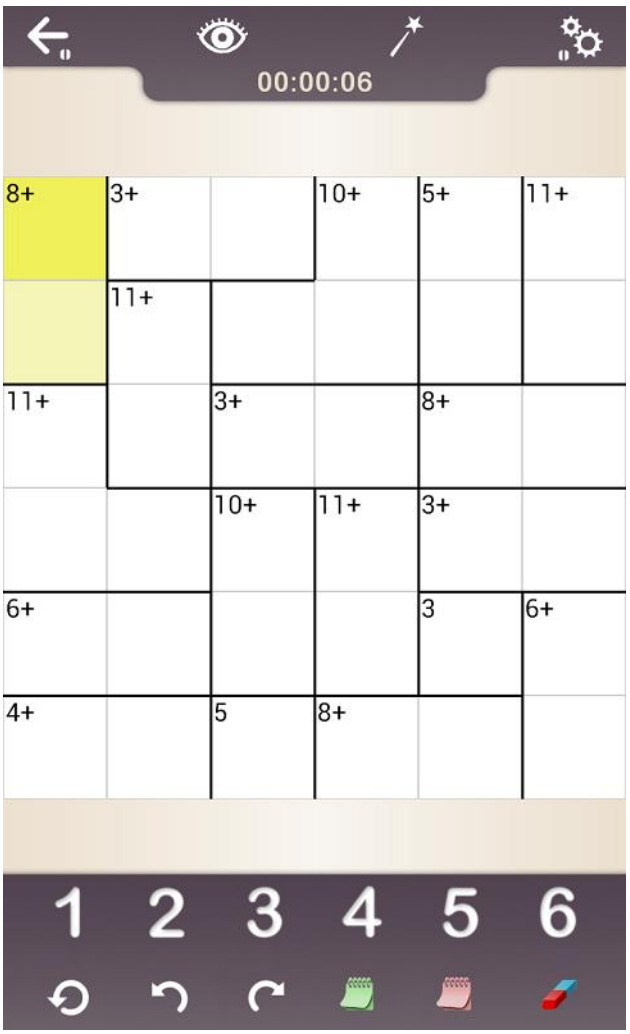

Рисунок 1.7 – KenKen

Головоломки KenKen — це корисна розвага для дітей, яку варто пропонувати регулярно. Їх можна налаштувати під різні рівні, що підходить для учнів із різними вміннями. Програма "KenKen Classroom" (KKCR) допомагає вчителям, надаючи готові набори головоломок для уроків. Ця гра дуже популярна — її публікують у The New York Times і багатьох інших виданнях у світі. Вілл Шортц, редактор головоломок в NY Times, називає KenKen "найцікавішою грою з часів судоку".

Додатки на телефон, такі як "KenKen Classic" і "KenKen Classic II", отримали багато хороших відгуків за свою простоту і захоплюючість. Але деякі користувачі скаржаться на проблеми, наприклад, неотримані головоломки після покупки в додатку або надокучливу рекламу.

Math Crossword від Math Game Time — це онлайн-гра, яка вимагає від гравців заповнювати числа та знаки для завершення рівнянь, закріплюючи базові математичні факти. Math Crossword — Number puzzle вказана як подібна гра до Crossmath, що вказує на ширшу категорію таких головоломок.

Kakuro прямо називають "математичним кросвордом" або "Крос-Суми". Гра Crossmath у своїй назві має фразу "математичні кросворди". Хоча KenKen прямо не називається кросвордом, її формат — сітка з числами та підказками, схожа на Судоку, — робить її частиною цієї категорії головоломок. Такий підхід до "кросворду" показує, що знайома форма гри спрощує і робить цікавими завдання з числами, перетворюючи складну математику на зрозумілі головоломки. Популярність цих ігор свідчить про те, що, використовуючи вже знайомі ігрові механіки (як у кросвордах) для математики, можна значно спростити їх освоєння. Завдяки цьому процес вирішення задач стає приємним і мотивуючим, перетворюючи складні завдання у захопливу гру. Це по суті "ігрифікує" саму математику.

Crossmath пропонує діапазон рівнів складності: "Легкий, Середній, Важкий та Експерт". KenKen дозволяє гравцям вибирати розміри сітки від 3x3 до 9x9 та різні налаштування складності. Kakuro також відзначається наявністю "легких аркушів головоломок" для формування впевненості. Крім того, такі функції, як "Необмежена кількість скасувань" та "Посилення" в Crossmath, а також "Підказки, Перевірка, Скасування" в KenKen, явно розроблені для підтримки гравця. Цей всебічний підхід до масштабування складності та внутрішньоігрової підтримки підкреслює складне розуміння дизайну навчання в цих іграх. Вони не просто пропонують проблеми, а й структурують процес навчання. Можливість регулювати рівень складності та надавати допомогу гарантує, що гравці різного рівня майстерності можуть залучатися без надмірного розчарування, сприяючи розвитку мислення зростання та стійкій залученості. Ця адаптивність є критичним фактором для максимізації освітньої ефективності та забезпечення широкої привабливості.

Популярні додатки Crossmath і KenKen отримують гарні оцінки, але багато користувачів негативно відгукуються про нав'язливу рекламу. Вона з'являється навіть тоді, коли телефон вимкнений, або коли гравці натискають "продовжити" без отримання бонусів. Це викликає сильне роздратування, і деякі люди навіть видаляють додаток. Розробники пояснюють, що їм потрібна реклама для заробітку, оскільки вони є "стартапом". Але занадто багато реклами може зіпсувати досвід користувачів і зменшити освітню цінність ігор. Батьки та вчителі повинні враховувати цей фактор, адже він впливає на мотивацію дітей і якість навчання. Додатки пропонують функцію платного видалення реклами, але через поганий перший досвід деякі користувачі можуть не захотіти цього робити.

Таблиця 1.1 – Порівняльний аналіз ключових математичних кросвордів

Назва гри	**Основні математичні концепції**	**Основні механіки**	**Цільова аудиторія**	**Платформи**	**Особливості / Відгуки**
Crossmath - Math Puzzle Games	Додавання, віднімання, множення, ділення, логічна дедукція, вирішення проблем	Сіткова, заповнення пропусків, завершення рівнянь, стратегічне заповнення чисел та операторів	Діти (7+), Дорослі, Загальна публіка	ПК, Android, iOS, Веб, Фізична настільна гра	10М+ завантажень, 4.8 зірки (Google Play), 4.5 зірки (App Store). Проблеми з надмірною/нав'язливою рекламою.
Kakuro (Крос-Суми)	Додавання, логічна дедукція, комбінаторне мислення	Сіткова, заповнення цифр (1-9), сума чисел у "слові" дорівнює підказці	Діти (8-12), Дорослі, Всі віки	Мобільні додатки (iOS, Android), Веб	"Більший хіт у Японії, ніж Судоку!", 4.5 зірки (Kakuro U), позитивні відгуки за логіку та відточування навичок.
KenKen	Арифметика (додавання, віднімання, множення, ділення), логічна дедукція, стратегічне мислення	Сіткова, заповнення цифр (1-9), без повторень у рядках/стовпцях, "клітки" з цільовими числами та операціями	Діти, Дорослі, Всі віки	ПК, Android, iOS, Веб	Щоденно в NYT, 4.7 зірки (iOS), 4.5 зірки (Google Play), використовується 30К+ вчителів. Проблеми з рекламою та покупками в додатку.

Тож, як ми бачимо, хоча вже створено багато програм, які реалізують математичні кросворди, в кожній з них ще є ті чи інші недоліки. Спільним недоліком можна вважати відсутність української локалізації та адаптивності до гравця.

З метою подолання цих недоліків, створення комп'ютерної гри, орієнтованої на український ринок, з можливістю адаптації до рівня гравця, ми і ставимо перед собою задачу розробити навчальну комп'ютерну гру-головоломку «Математичні кросворди».

1.3 Визначення вимог до навчальної комп'ютерної гри-головоломки «Математичні кросворди»

Бачення "Математичних Кросвордів" полягає у створенні високозахопливої та ефективної освітньої гри-головоломки, яка зробить вивчення математики приємним та доступним для широкого кола користувачів, особливо для дітей та учнів. Поєднуючи знайому структуру кросвордів з числовими завданнями, гра має на меті розвивати навички усного рахунку, логічної дедукції та стратегічного мислення у веселому, інтерактивному середовищі. Вона слугуватиме додатковим навчальним інструментом, що закріплює концепції, вивчені в класі, та заохочує самостійний розвиток навичок.

Основні Механіки Геймплею та Джерела Натхнення

Гра черпатиме натхнення з усталених числових кросвордів, таких як Какуро та КенКен, які довели свою освітню цінність.

Какуро (Cross Sums): Основна механіка Какуро полягає у заповненні сітки цифрами від 1 до 9 таким чином, щоб кожне горизонтальне та вертикальне "слово" сумувалося до заданої підказки, при цьому цифри не повинні повторюватися в межах одного "слова". Цей підхід безпопосередньо відповідає концепції "Математичних Кросвордів", акцентуючи увагу на додаванні та логічній дедукції.

КенКен: Ця сіткова головоломка вимагає заповнення клітинок цифрами (наприклад, 1-4 для сітки 4x4, 1-6 для 6x6) таким чином, щоб жодна цифра не повторювалася в жодному рядку чи стовпці (властивість латинського квадрата), а числа в "клітках" (виділених жирним контуром групах клітинок) утворювали задане "цільове" число за допомогою вказаних математичних операцій (додавання, віднімання, множення, ділення). Включення КенКену з його різними операціями та концепцією "кліток" пропонує значну гнучкість для розробки головоломок та розвитку математичних навичок, виходячи за межі лише додавання.

Кросматематика (Crossmath): Існуючі ігри "Кросматематика" вже поєднують елементи кросвордів та математичних головоломок, зосереджуючись на додаванні, відніманні, множенні та діленні для розв'язання рівнянь. Ці ігри часто мають різні рівні складності (Легкий, Середній, Важкий, Експертний), щоденні завдання та нескінченні режими.

Запропоноване поєднання для "Математичних Кросвордів": Гра інтегруватиме правила заповнення сітки та сумування "слів" Какуро з логікою "кліток" КенКену, що використовує кілька операцій. Аспект "кросворду" проявлятиметься у структурі сітки та представленні підказок, де підказки "по горизонталі" та "по вертикалі" визначатимуть числові суми або операції для пересічних клітинок.

Ключовим нововведенням є те, що, на відміну від традиційного Какуро, де підказки є лише сумами, "Математичні Кросворди" представлятимуть "підказки" як арифметичні вирази (наприклад, "x + y = 10", "a * b = 24") або цільові числа для багатокоміркових "кліток" (як у КенКен). Це дозволить охопити ширший спектр математичних концепцій та типів задач. Головоломки будуть генеруватися за допомогою надійних алгоритмів, що створюватимуть унікальні та розв'язні завдання, змінюючи розміри сітки (наприклад, від 3x3 до 9x9, подібно до КенКен) та рівні складності.

Взаємозв'язок між елементами головоломки є фундаментальним для її освітньої цінності. Аналогія з кросвордом виходить за межі простого

візуального макета; вона охоплює взаємозалежність підказок. У традиційному текстовому кросворді розв'язання однієї підказки допомагає розв'язати пересічні. Подібним чином, у Какуро та КенКен заповнення однієї клітинки впливає на можливості в її рядку, стовпці та клітці. Це означає, що алгоритм генерації головоломок повинен забезпечувати наявність цієї взаємозалежності та її розв'язність за допомогою логічної дедукції, а не просто грубої сили. Це також означає, що користувацький інтерфейс (UI) повинен чітко висвітлювати ці пересічні зв'язки. Ця взаємозалежність є основною педагогічною перевагою, оскільки вона змушує гравців мислити цілісно та застосовувати логічну дедукцію до кількох обмежень одночасно, що є навичкою мислення вищого порядку, ніж просте арифметичне тренування. Це не просто "математика в сітці", а "математика через взаємопов'язану логіку сітки".

Хоча КенКен був розроблений як "метод тренування мозку без інструкцій" , а правила Какуро "прості та легкі для розуміння" , освітній контекст вимагає певної підтримки. Дослідження підкреслюють важливість "легких аркушів головоломок" для формування впевненості , пояснення помилок та надання підказок. Це вказує на делікатний баланс: гра повинна бути інтуїтивно зрозумілою та доступною на базовому рівні, але мати надійні педагогічні системи підтримки (навчальні посібники, прогресивну складність, зворотний зв'язок щодо помилок, підказки) для забезпечення ефективного навчання та запобігання розчаруванню, особливо для цільової аудиторії дітей. Це поєднання "безінструктивного" підходу для початкового залучення та "підтримуваного навчання" для сталого освітнього зростання. Така подвійність впливає на дизайн UI/UX (чистий інтерфейс для основної гри, але доступні накладки довідки/навчання), алгоритми адаптивної складності (не просто збільшення чисел, а введення нових складнощів правил або зменшення підказок) та стратегію вмісту (структуроване введення механік та прогресивних завдань).

Властивість "майже нескінченної" генерації головоломок є ключовою перевагою для освітньої цінності та утримання гравців. Головоломки Какуро

"майже нескінченні та неможливі для завершення навіть за сотні життів" через комбінації розмірів кліток, можливостей та сум. КенКен також пропонує "практично необмежену кількість головоломок". Ця "майже нескінченна" властивість є не просто функцією; вона є критичним фактором для тривалого освітнього впливу та утримання гравців. Для навчальної гри повторення з варіаціями є ключем до майстерності. Кінцевий набір головоломок призводить до запам'ятовування, а не до справжнього розвитку навичок. Нескінченні головоломки забезпечують постійні нові завдання, запобігаючи нудьзі та надаючи нескінченні можливості для практики. Це безпосередньо підтримує аспект "подорожі, а не пункту призначення" майстерності в адаптивній складності. Це підкреслює першочергову важливість алгоритму генерації головоломок як основної технічної вимоги. Він повинен бути висококонфігурованим (розмір сітки, операції, складність), щоб повною мірою використовувати цей "нескінченний" потенціал для адаптивного навчання та довгострокового залучення. Це також означає меншу потребу в постійних оновленнях вмісту, переносячи фокус розробки на вдосконалення алгоритмів та додавання функцій.

Освітні цілі та цільова аудиторія

Первинні освітні цілі:

Арифметична вправність: Покращення швидкості та точності в базових операціях (додавання, віднімання, множення, ділення).

Логічна дедукція та критичне мислення: Покращення навичок розв'язання проблем, абстрактного мислення та стратегічного мислення.

Числове чуття: Розвиток глибшого розуміння числових взаємозв'язків та комбінацій.

Стійкість до розв'язання проблем: Заохочення наполегливості у вирішенні складних завдань.

Цільова аудиторія: Гра буде орієнтована насамперед на дітей віком 7-12 років, як це пропонується придатністю Какуро та Кросматематики для цієї вікової групи. Однак, завдяки регульованим рівням складності , вона також

може бути цікавою для старших учнів та дорослих, які шукають тренування мозку та розважальні математичні головоломки. Освітня цінність таких ігор добре задокументована, демонструючи значне покращення здібностей до розв'язання математичних та наукових задач.

Педагогічні принципи залучення та навчання

Активне навчання: Формат головоломки за своєю суттю сприяє активній участі, вимагаючи від гравців аналізувати, розробляти стратегії та виводити рішення, а не пасивно отримувати інформацію.

Миттєвий зворотний зв'язок: Надання миттєвого зворотного зв'язку щодо правильних/неправильних введень є вирішальним для навчання та залучення. Це допомагає гравцям зрозуміти помилки та швидко скоригувати стратегії.

Адаптивна складність: Гра повинна динамічно регулювати складність залежно від продуктивності гравця, щоб підтримувати стан "потоку", запобігаючи нудьзі (занадто легко) або розчаруванню (занадто важко). Це забезпечує постійний виклик та освоєння матеріалу.

Формування впевненості: Легкі головоломки та можливості для успіху є життєво важливими, особливо для молодших учнів, щоб сформувати впевненість та розпалити дух змагання.

Обробка помилок та керівництво: Замість покарання за помилки, гра повинна пропонувати чіткі пояснення помилок та надавати підказки або вказівки, коли гравець стикається з труднощами, не розв'язуючи всю головоломку за нього. Функція необмеженого скасування є цінною.

Мотивація та винагороди: Включення таких елементів, як бали, значки, таблиці лідерів та тематичні події, для підвищення мотивації та відстеження прогресу.

Спільна гра (необов'язково): Хоча гра в основному розрахована на одного гравця, дизайн може передбачати функції, що сприяють спільному розв'язанню проблем, оскільки КенКен припускає, що робота в парах може допомогти учням побачити нові стратегії та зменшити тиск.

Основні вимоги до програмного забезпечення

Система сітки:

- Динамічна генерація сітки, що підтримує різні розміри (наприклад, від 3x3 до 9x9) для розміщення різних рівнів складності та прогресу гравця.
- Клітинки, здатні містити одноцифрові числа (1-9).
- Підтримка "кліток" або "регіонів" (подібно до КенКен) з відповідними цільовими числами та математичними операціями (+, -, ×, ÷).
- Можливість визначати "слова" (рядки/стовпці) з цільовими сумами (подібно до Какуро).

Забезпечення правил:

- Запобігання повторенню цифр у "словах" у стилі Какуро (рядках/стовпцях).
- Запобігання повторенню цифр у рядках/стовпцях у стилі КенКен (властивість латинського квадрата).
- Перевірка операцій у клітках та цільових чисел.

Алгоритм генерації головоломок:

- Процедурна генерація унікальних, розв'язних головоломок для різних розмірів сітки та рівнів складності.
- Алгоритм повинен забезпечувати єдине логічне рішення для кожної головоломки.

Настроювані параметри складності, включаючи:

- Розмір сітки.
- Кількість та складність операцій (базова арифметика, потенційно розширені, такі як піднесення до степеня/модуль для експертних рівнів).
- Кількість заданих "вільних місць" або попередньо заповнених клітинок.

- Розмір та форма кліток.
- Діапазон використовуваних чисел (наприклад, 1-9, або потенційно включаючи 0 або від'ємні значення для просунутих рівнів).
- Можливість генерувати головоломки "Кросматематика", які поєднують кросворди та математичні задачі.

Механізми введення:

- Інтуїтивне введення для розміщення цифр у клітинках (наприклад, цифрова клавіатура, колесо вибору).
- Функціональність "нотаток" або "олівцевих позначок", що дозволяє гравцям записувати можливі варіанти для клітинок.

Управління станом гри:

- Збереження/завантаження прогресу гри, включаючи кілька одночасних головоломок.
- Необмежена функція скасування/повтору.
- Функція паузи.

Взаємодія генерації головоломок та адаптивної складності є технічною основою освітньої ефективності. Алгоритм адаптивної складності має безпосередньо впливати на параметри генерації головоломок. Якщо гравець відчуває труднощі, генератор повинен створювати легшу головоломку (наприклад, менша сітка, менше операцій, більше попередньо заповнених клітинок). Якщо гравець досягає успіху, генератор повинен збільшувати складність. Цей динамічний цикл зворотного зв'язку (продуктивність гравця -> адаптивний алгоритм -> генерація головоломок -> нова головоломка -> продуктивність гравця) створює стан "потоку" та сприяє майстерності. Без висококонфігурованого генератора адаптивна складність обмежена простими підказками або коригуванням балів. Це означає, що пріоритет розробки повинен бути на бездоганній спільній роботі цих двох систем. Їхні точки інтеграції та механізми обміну даними є критичними аспектами дизайну. Це також свідчить про те, що "рівні складності" (Легкий, Середній, Важкий)

повинні слугувати початковими базовими рівнями, а адаптивна система повинна точно налаштовуватись у межах цих категорій та між ними.

Користувацький інтерфейс (UI) та користувацький досвід (UX)

Чіткість та інтуїтивність:

- Чистий, незахаращений інтерфейс, що надає пріоритет видимості головоломки.
- Чітке візуальне розрізнення кліток, підказок та редагованих клітинок.
- Інтуїтивні елементи керування для введення цифр та створення нотаток.

Візуальний зворотний зв'язок:

- Миттєвий візуальний зворотний зв'язок щодо правильних/неправильних введень.
- Виділення активного рядка/стовпця/клітки.
- Візуальні підказки для помилок повторення цифр.

Функції доступності (Відповідність WCAG 2.1):

Сприйнятливість:

- Великі шрифти та опції темного режиму для кращого перегляду та зменшення напруги очей.
- Альтернативний текст для зображень та нетекстового вмісту.
- Інформація, що передається через презентацію (наприклад, помилки, прогрес), повинна бути програмно визначуваною або доступною у текстовому вигляді.
- Уникнення покладання виключно на колір для зворотного зв'язку (наприклад, зелений для правильного, жовтий для неправильно розміщеного, як у Wordle – забезпечення альтернативних індикаторів).

Керованість:

- Повна підтримка навігації за допомогою клавіатури.

- Альтернативні методи введення (наприклад, дотик, миша, клавіатура).
- Достатній час для читання та використання вмісту; регульований час або відсутність часових обмежень, де це можливо.
- Мінімізація або надання опцій для анімації від взаємодій.
- Розміри цільових елементів для інтерактивних елементів повинні бути достатньо великими для легкої взаємодії.

Зрозумілість:

- Чіткі інструкції та повідомлення щодо ігрових механік та правил.
- Послідовна навігація та елементи інтерфейсу.

Надійність:

- Сумісність з різними користувацькими агентами та допоміжними технологіями.

Естетика:

- Приємна колірна палітра та добре продуманий макет.
- Настроювані теми/колірні схеми.
- Додаткові заспокійливі звукові ефекти та музика.

Продуктивність:

- Плавні анімації та переходи.
- Швидкий час завантаження.

Чуйний інтерфейс на різних пристроях та розмірах екрану.

Для навчальної гри дизайн UI/UX виходить за межі простої зручності використання; він безпосередньо впливає на ефективність навчання та утримання гравців. Поганий інтерфейс (наприклад, дрібні шрифти, нечутливі елементи керування) збільшує когнітивне навантаження, відволікаючи розумову енергію від самого завдання розв'язання головоломки. Елементи, що викликають розчарування (наприклад, нав'язлива реклама, відсутність функції скасування), призводять до втрати зацікавленості та видалення гри, що прямо суперечить освітній меті постійної практики. Акцент на "навчанні без

інструкцій" означає, що сам інтерфейс повинен неявно навчати правилам. Функції доступності є не просто відповідністю стандартам, а й необхідними для забезпечення того, щоб всі учні могли ефективно залучатися, безпосередньо підтримуючи освітню місію гри щодо інклюзивності. Це підвищує дизайн UI/UX від "бажаного" до "критичного фактора успіху" для навчальної гри. Це вимагає широкого тестування користувачів з цільовою демографічною групою (дітьми) для виявлення та зменшення точок тертя. Процес дизайну повинен бути ітеративним, включаючи зворотний зв'язок щодо зручності використання та результатів навчання.

Системи Прогресу Гравця та Залучення

Рівні складності:

- Попередньо встановлені рівні складності (Легкий, Середній, Важкий, Експертний).
- Поступове введення нових механік та підвищення складності.
- Рівні для відпочинку після складних завдань.

Алгоритм адаптивної складності:

- Збір даних: Відстеження показників гравця, таких як точність, час виконання, кількість використаних підказок, кількість скасувань та показники завершення головоломок.
- Розробка алгоритму: Впровадження динамічного алгоритму (наприклад, зваженого середнього) для коригування складності на основі зібраних даних.

Механізми коригування:

- Динамічне коригування складності головоломки (наприклад, розмір сітки, кількість клітинок у клітках, складність операцій, кількість попередньо заповнених клітинок).
- Коригування частоти/доступності підказок або бонусів.
- Тонке коригування ігрових параметрів для підтримки "потоку" без надмірної помітності.

- Тестування та ітерація: Постійне тестування та вдосконалення адаптивної системи на основі зворотного зв'язку гравців та даних про продуктивність.

Мотивація та винагороди:

- Відстеження статистики: Детальні записи геймплею, моніторинг прогресу та високі бали.
- Таблиці лідерів: Глобальні або дружні таблиці лідерів для конкурентоспроможних гравців.
- Досягнення/Значки: Значки, що розблоковуються за завершення тематичних подій або досягнення етапів.
- Щоденні завдання: Заохочення до регулярної гри та підтримки навичок.
- Тематичні події/Пригоди: Обмежені за часом події для перевірки логічних навичок та розблокування спеціальних винагород.
- "Нескінченний режим": Режим, де помилки не перевіряються до остаточного подання, що заохочує стратегічне мислення для отримання вищих балів.

Таблиця 1.2 – Метрики Адаптивної Складності та Приклади Коригування

Метрика	**Інтервал Вимірювання**	**Індикація (що свідчить про навичку/боротьбу гравця)**	**Приклад Коригування (Збільшення Складності)**	**Приклад Коригування (Зменшення Складності)**
Час виконання головоломки	За головоломку, за сесію, середнє значення	Швидкість розв'язання, вправність	Зменшення часу на головоломку, збільшення розміру сітки	Збільшення часу на головоломку, зменшення розміру сітки
Показник точності	За головоломку, за сесію	Розуміння правил, уважність	Зменшення кількості попередньо заповнених клітинок	Збільшення кількості попередньо заповнених клітинок
Кількість використаних підказок	За головоломку	Потреба в допомозі, складність головоломки	Зменшення доступності підказок	Збільшення доступності підказок, надання більш

				детальних підказок
Кількість скасувань	За головоломку	Схильність до помилок, експериментування	Збільшення складності операцій, зменшення кліток з одним рішенням	Зменшення складності операцій, надання "дихальних" рівнів
Послідовні успіхи/невдачі	За сесію	Підтвердження рівня майстерності, потреба в адаптації	Перехід до наступного рівня складності	Перехід до попереднього рівня складності, повторення легших головоломок

2 ПРОЕКТУВАННЯ НАВЧАЛЬНОЇ КОМП'ЮТЕРНОЇ ГРИ-ГОЛОВОЛОМКИ «МАТЕМАТИЧНІ КРОСВОРДИ»

2.1 Моделювання гри «Математичні кросворди»

Діаграма варіантів використання (Use Case Diagram) створюється, що проілюструвати взаємодію між користувачами (акторами) та системою, показуючи основні функції, які система надає.

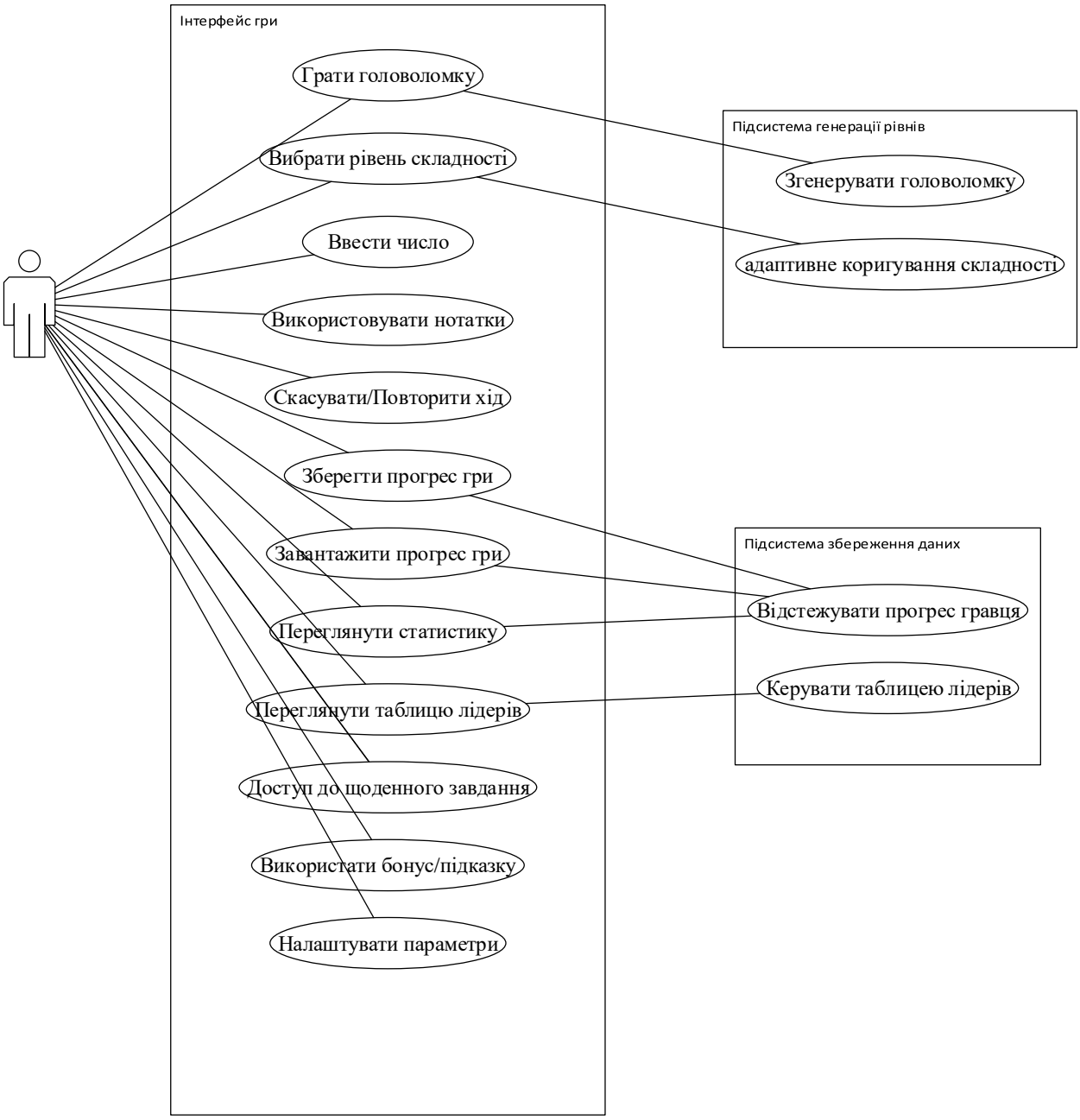

Рисунок 2.1 – Діаграма використання гри

Гравець (Player): Основний актор, який взаємодіє з грою. Він може грати головоломки, вибирати рівень складності, вводити числа, використовувати нотатки, скасовувати/повторювати ходи, зберігати/завантажувати прогрес, переглядати статистику та таблицю лідерів, отримувати доступ до щоденних завдань та тематичних подій, використовувати бонуси та здійснювати покупки.

Система (System): Представляє саму гру та її внутрішні процеси. Вона відповідає за генерацію головоломок, перевірку введення, відстеження прогресу, керування таблицею лідерів, монетизацію, застосування адаптивної складності та надання зворотного зв'язку.

Варіанти використання (Use Cases):

Основні ігрові функції: Грати головоломку, Ввести число, Використовувати нотатки/позначки олівцем, Скасувати/Повторити хід.

Управління грою: Вибрати рівень складності, Зберегти прогрес гри, Завантажити прогрес гри, Налаштувати параметри.

Прогрес та залучення: Переглянути статистику, Переглянути таблицю лідерів, Доступ до щоденного завдання, Брати участь у тематичній події, Використати бонус/підказку.

Системні взаємодії: Отримати зворотний зв'язок, Отримати адаптивне коригування складності (ці варіанти використання розширюють основний ігровий процес).

Відносини:

Асоціації: Зв'язують акторів з варіантами використання, в яких вони беруть участь.

Включення (<<include>>): Вказує, що один варіант використання включає функціональність іншого. Наприклад, Грати головоломку включає Згенерувати головоломку.

Розширення (<<extend>>): Вказує, що один варіант використання може розширювати функціональність іншого за певних умов.

Діаграма класів (Class Diagram)

Діаграма класів надає статичний, структурний вигляд системи, показуючи класи, їхні атрибути, методи та взаємозв'язки.

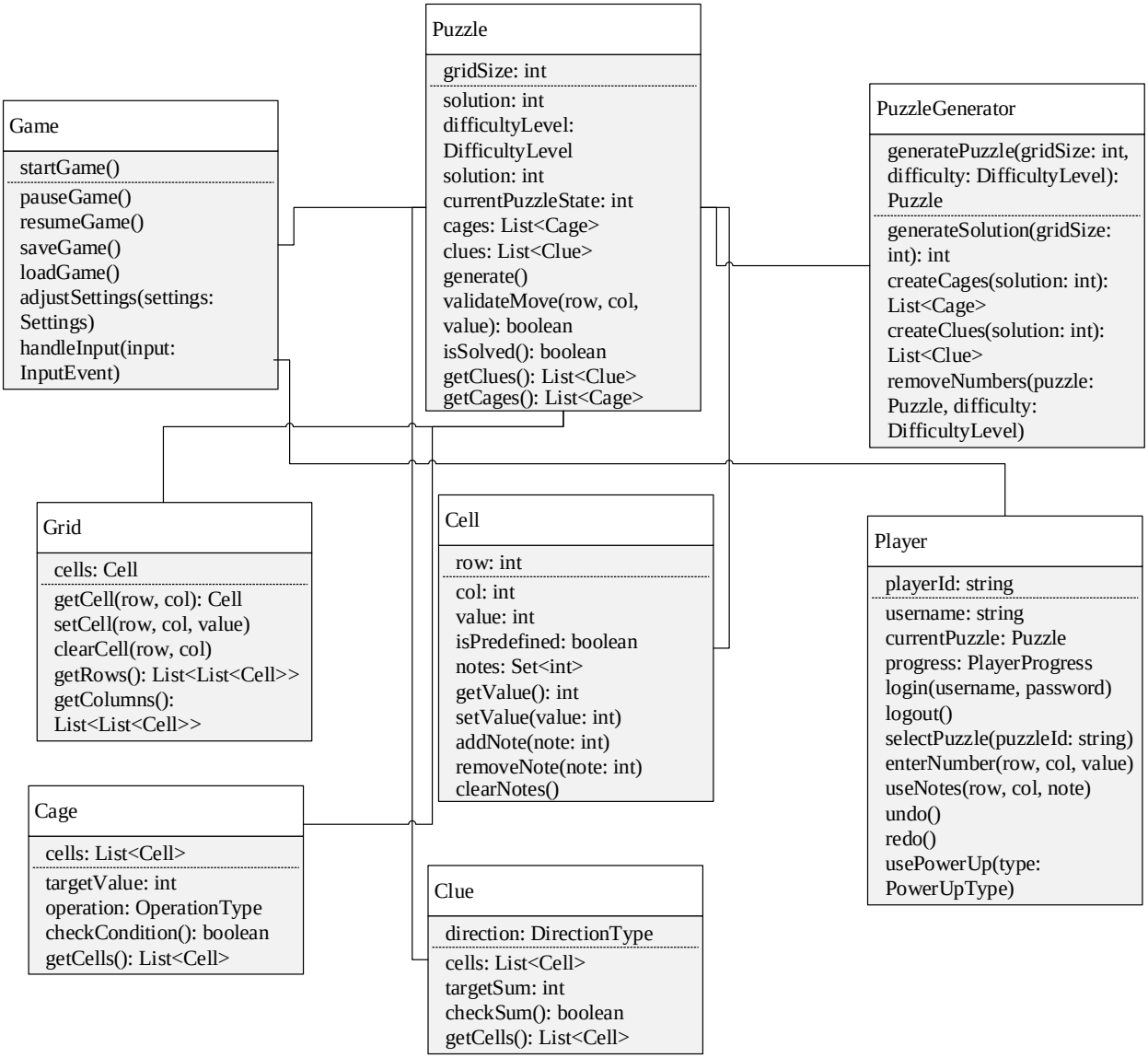

Рисунок 2.2 – Діаграма класів

Опис діаграми класів:

Game: Центральний клас, який керує загальним станом гри, ігровим циклом та взаємодією між іншими компонентами.

Puzzle: Представляє одну головоломку. Містить сітку, клітки (для КенКен) та підказки (для Какуро/Кросматематики). Відповідає за перевірку розв'язності та стану головоломки.

Grid: Представляє ігрове поле, що складається з об'єктів Cell.

Cell: Окрема клітинка на сітці. Зберігає значення, чи є вона попередньо визначеною, та нотатки гравця.

Cage: Представляє "клітку" (регіон) у стилі КенКен. Містить список клітинок, цільове значення та математичну операцію.

Clue: Представляє підказку для "слова" у стилі Какуро (сума для рядка або стовпця). Містить список клітинок та цільову суму.

PuzzleGenerator: Відповідає за процедурну генерацію нових, унікальних та розв'язних головоломок на основі заданих розмірів та рівня складності.

Player: Представляє поточного гравця. Керує діями гравця, такими як введення чисел, використання нотаток, скасування/повторення.

PlayerProfile: Зберігає постійні дані гравця, включаючи налаштування та досягнення.

StatisticsManager: Збирає та керує статистикою гравця (час, помилки, завершення головоломок).

LeaderboardManager: Керує глобальними та дружніми таблицями лідерів.

SettingsManager: Керує налаштуваннями гри (наприклад, великі шрифти, темний режим).

AdaptiveDifficultyManager: Аналізує продуктивність гравця та динамічно коригує рівень складності наступних головоломок.

MonetizationManager: Обробляє внутрішньоігрові покупки, видалення реклами та показ реклами з винагородою.

InputHandler: Обробляє введення користувача (дотик, клавіатура, миша) та передає його відповідним компонентам.

UIManager: Керує відображенням гри, оновленнями інтерфейсу та візуальним зворотним зв'язком.

SoundManager: Керує відтворенням звукових ефектів та музики.

Перерахування (Enums): DifficultyLevel, OperationType, DirectionType надають визначені набори значень для відповідних атрибутів.

Ці діаграми надають чітке візуальне представлення структури та функціональності програмного забезпечення "Математичних Кросвордів", що є важливим для подальшої розробки.

2.2 Проєктування основних алгоритмів

Як Какуро, так і КенКен є класичними прикладами задач із задоволенням обмежень (CSP). CSP визначається набором змінних, кожна з яких має область можливих значень, та кінцевим набором обмежень, що обмежують значення, які можуть приймати ці змінні. Метою є присвоєння значень усім змінним таким чином, щоб усі обмеження були задоволені.

У контексті цих головоломок:

Змінні: Окремі білі клітинки (вхідні клітинки) у сітці.

Домени: Початковий домен для кожної змінної (клітинки) зазвичай є набором однозначних чисел від 1 до 9 (для Какуро) або від 1 до N (для КенКен, де N — ширина/висота сітки).

Обмеження:

Обмеження унікальності:

Какуро: Жодна цифра не може повторюватися в межах одного горизонтального або вертикального "пробігу".

КенКен: Жодна цифра не може повторюватися в жодному рядку або стовпці (обмеження латинського квадрата).

Арифметичні обмеження:

Какуро: Сума чисел у кожному "пробігу" повинна дорівнювати відповідній підказці.

КенКен: Числа в кожній "клітці" повинні давати вказане цільове число за допомогою заданої математичної операції (+, -, ×, ÷).

Обмеження діапазону: Усі призначені цифри повинні знаходитися в межах зазначеного діапазону (наприклад, 1-9).

Класифікація Какуро як NP-повної задачі та використання алгоритму пошуку з поверненням (backtracking search) разом із поширенням обмежень

для розв'язання та генерації головоломок вказують на те, що це обчислювально інтенсивні проблеми. NP-повнота означає, що пошук рішення або генерація унікальної головоломки може стати експоненційно затратним за часом зі збільшенням розміру сітки та складності. Це свідчить про те, що для практичних застосувань, особливо на пристроях з обмеженими ресурсами, таких як мобільні телефони, ефективність алгоритму має першочергове значення. Принцип "fail-first" у CSP, який спрямований на якомога раннє відсікання дерева пошуку шляхом прийняття рішень щодо змінних з найменшою кількістю варіантів, є прямою відповіддю на цей обчислювальний виклик. Це означає, що розробка алгоритму повинна надавати пріоритет високо оптимізованому поширенню обмежень та інтелектуальним евристикам впорядкування змінних/значень, щоб керувати притаманною "складністю" та забезпечувати прийнятний час генерації головоломок, тим самим покращуючи користувацький досвід та зменшуючи навантаження на пристрій.

2.2.1 Детальний алгоритм пошуку з поверненням для унікальності рішення

Основний принцип: Пошук з поверненням є найпоширенішим та найефективнішим алгоритмом для розв'язання та генерації CSP, таких як Судоку та КенКен. Він працює шляхом поступового побудови рішення, і якщо часткове рішення призводить до протиріччя (порушує обмеження), він "повертається" до останньої точки прийняття рішення та пробує альтернативний шлях.

Процес генерації (загальні кроки для унікального рішення):

Генерація повної, дійсної сітки-рішення:

Почати з порожньої сітки.

Використовувати рандомізований розв'язувач з поверненням для заповнення всієї сітки, забезпечуючи дотримання всіх правил, специфічних для головоломки (наприклад, латинський квадрат для КенКен, або початкові обмеження суми для Какуро). Рандомізація у виборі значень або порядку

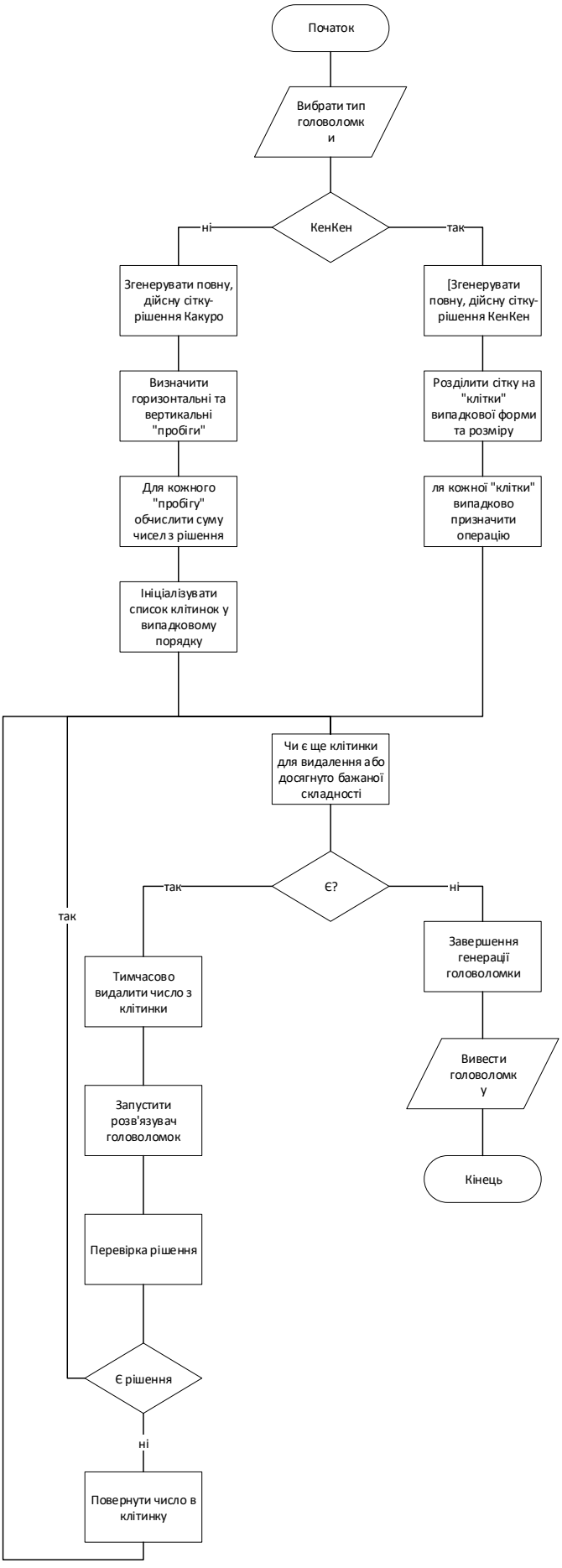

Рисунок 2.3 – Блок-схема алгоритму генерації головоломок

клітинок забезпечує унікальне початкове рішення кожного разу. Для КенКен це передбачає генерацію випадкового латинського квадрата як рішення.

Визначення кліток/пробігів та розрахунок підказок:

КенКен: Розділити повністю розв'язану сітку на "клітки". Це можна досягти шляхом створення неорієнтованого графа, де вершини є клітинками, а ребра з'єднують сусідні клітинки, потім випадково розбити цей граф на зв'язні компоненти, обмежуючи кожну компоненту випадково обраним максимальним розміром. Для кожної клітки випадково призначити арифметичний оператор (+, -, ×, ÷) та обчислити відповідне цільове значення з чисел у розв'язаній сітці. Одноклітинні клітки просто приймають значення клітинки як цільове.

Какуро: Визначити всі горизонтальні та вертикальні "пробіги" білих клітинок. Для кожного пробігу обчислити суму чисел з розв'язаної сітки, щоб вона слугувала підказкою.

Ітеративне видалення клітинок та перевірка унікальності:

Це вирішальний крок для створення головоломки з унікальним рішенням та контролю складності.

Відвідувати клітинки в сітці у випадково переставленому порядку.

На кожній клітинці тимчасово стерти її номер.

Запустити розв'язувач головоломок (також заснований на пошуку з поверненням та поширенні обмежень) на зміненій сітці, щоб перевірити, чи має вона все ще унікальне рішення. Це зазвичай включає пошук одного рішення, потім заборону його та пошук іншого; якщо друге рішення не знайдено, воно унікальне.

Якщо видалення призводить до кількох рішень, число повертається на місце. Якщо воно залишається унікальним, число залишається видаленим.

Повторювати цей процес до досягнення бажаної кількості порожніх клітинок (або мінімальної кількості підказок, наприклад, 17 для головоломок типу Судоку), або доки не можна буде видалити більше клітинок, зберігаючи

унікальність. Кількість спроб видалити значення також може впливати на складність.

Поширення обмежень (інтерлейтинг з пошуком з поверненням): Ця техніка є життєво важливою для ефективності, відсікаючи простір пошуку шляхом усунення значень з доменів клітинок, які несумісні з поточними призначеннями або обмеженнями. Вона працює шляхом постійного зменшення набору можливих значень для кожної нерозв'язаної клітинки на основі вже розміщених значень та правил головоломки.

Приклади правил поширення:

Коли клітинка розв'язана, видалити її значення з доменів інших клітинок у тому ж рядку та стовпці (для КенКен) або в тому ж пробігу (для Какуро).

Якщо клітка (КенКен) або пробіг (Какуро) має лише одну можливу комбінацію чисел, яка задовольняє її підказку, ці числа присвоюються відповідним клітинкам.

Якщо рядок, стовпець або пробіг має лише одну клітинку, куди можна помістити певне значення, це значення присвоюється цій клітинці.

Визначити "самостійні блоки" (підголоволомки, які можна розв'язати незалежно) та звести їх до єдиного визначеного значення, спрощуючи більшу головоломку.

Застосовувати логіку "превентивних наборів" для видалення значень з інших клітинок у векторі, якщо підмножина клітинок повинна містити певний набір значень.

Процес генерації головоломок значною мірою залежить від випадковості (наприклад, початкове заповнення сітки, розбиття кліток, порядок видалення клітинок). 1 Однак ця випадковість критично обмежується вимогою унікального рішення 2 та контрольованої складності. 3 Це означає делікатний баланс: неконтрольована випадковість може призвести до нерозв'язних головоломок, тривіальних головоломок або головоломок з кількома рішеннями, що підриває освітню цінність та задоволення гравця. Основна думка полягає в тому, що надійний етап перевірки (перевірка

унікальності) діє як вирішальний детерміністичний фільтр для випадкової генерації, забезпечуючи якість та цілісність головоломок. Крім того, ефективне поширення обмежень 4 та інтелектуальні евристики (такі як "fail-first" з 5) стосуються не лише обчислювальної ефективності, а й спрямування випадкового пошуку до дійсних, унікальних та добре сформованих конфігурацій головоломок. Це свідчить про те, що сприйнята якість та освітня ефективність згенерованої головоломки є прямою функцією складності та взаємодії між компонентами випадкової генерації та детерміністичними механізмами задоволення/перевірки обмежень.

2.2.2 Механізми генерації сітки та присвоєння підказок

Алгоритм повинен спочатку визначити розміри сітки (наприклад, від 3x3 до 9x9 для КенКен , або змінні розміри для Какуро). Сітка логічно складатиметься з білих клітинок (куди поміщаються числа) та чорних клітинок (які слугують роздільниками або містять підказки).

Розміщення та форматування підказок:

Какуро: Чорні клітинки матимуть діагональну риску, з підказкою по горизонталі (сума для горизонтального пробігу) у верхньому правому куті та підказкою по вертикалі (сума для вертикального пробігу) у нижньому лівому куті. Чисто чорні клітинки використовуються для розділення пробігів.

КенКен: Клітки визначаються жирними контурами навколо груп клітинок, а цільове число та операція з'являються у верхньому лівому куті клітки. Алгоритм повинен генерувати ці межі кліток, забезпечуючи їх формування дійсних суміжних областей.

Генерація числових комбінацій (специфічно для Какуро): Алгоритм повинен мати механізм для ефективного визначення всіх унікальних наборів цифр (1-9, без повторень), які сумуються до заданої підказки для певної кількості клітинок. Наприклад, для суми 6 у 3 клітинках єдиним унікальним розбиттям є {1, 2, 3}. Попередній розрахунок або динамічне обчислення цих унікальних розбиттів (часто званих "комбінаціями" або "сумами") є

вирішальним як для генерації дійсних підказок, так і для ефективності розв'язувача.

Обробка операцій (специфічно для КенКен): Алгоритм повинен вміти обробляти чотири основні арифметичні операції. Для віднімання та ділення КенКен зазвичай обмежує підказки клітками з двох клітинок, щоб уникнути проблем з неасоціативністю (наприклад, 6 / 2 / 1 може бути (6/2)/1 = 3 або 6/(2/1) = 3). Алгоритм повинен або дотримуватися цього обмеження, або реалізувати складнішу логіку для керування відніманням/діленням з кількома клітинками.

Фізичне розташування сітки (розмір, розміщення чорних клітинок для Какуро або форма та розмір кліток для КенКен) є не просто візуальним елементом; воно принципово визначає типи та складність математичних задач, які можуть бути згенеровані в головоломці. Наприклад, довжина пробігу Какуро безпосередньо обмежує можливі унікальні розбиття , тоді як конфігурація кліток КенКен впливає на числові комбінації та операції, які можуть бути сформовані. Це означає, що складний алгоритм генерації повинен не випадковим чином розміщувати підказки або визначати клітки без врахування основних математичних наслідків. Натомість, він повинен спочатку інтелектуально будувати топологію сітки, потім заповнювати її числами (рішенням), а потім виводити підказки з цього розв'язаного стану, забезпечуючи, щоб отримані математичні задачі були дійсними, розв'язуваними та відповідали бажаним рівням складності. Концепції "розбиття квадрата на клітки" та ідентифікації "самостійних блоків" є яскравими прикладами того, як структурні властивості сітки використовуються для спрощення математичної задачі та підвищення ефективності генерації, підкреслюючи, що топологічні міркування є фундаментальними для ефективної та результативної генерації головоломок.

2.2.3 Динамічне масштабування та контроль складності

Важливість: Адаптивна складність є вирішальною для захоплюючої, інклюзивної та ефективної освітньої гри. Вона гарантує, що гравці постійно

стикаються з викликами, не відчуваючи себе перевантаженими або нудьгуючими, тим самим підтримуючи мотивацію та сприяючи відчуттю майстерності. Цей підхід є "етичним імперативом" для інклюзивності, забезпечуючи, щоб кожен міг відчути радість від оволодіння навичками.

Метрики для адаптації: Алгоритм повинен безперервно відстежувати різні показники продуктивності гравця для інформування про коригування складності:

Точність: Кількість правильних відповідей проти помилок.

Час завершення: Як швидко гравець розв'язує головоломку.

Використання ресурсів: Частота використання підказок, споживання посилень або кількість скасувань.

Показники успіху/невдачі: На конкретних типах головоломок, операціях або рівнях складності.

Показники когнітивного навантаження: Хоча явно не згадується в джерелах, це можна вивести з часу, витраченого на конкретні типи проблем, або кількості помилок.

Механізми коригування: Алгоритм може динамічно змінювати параметри головоломки:

Складність головоломки: Коригувати розмір сітки (наприклад, починаючи з 3x3 або 4x4 для початківців і переходячи до 9x9). Змінювати діапазон чисел або складність задіяних операцій (наприклад, введення множення/ділення в КенКен або сум з меншою кількістю унікальних розбиттів у Какуро). Збільшувати кількість порожніх клітинок (менше підказок) для підвищення складності.

Система підказок: Надавати частіші або детальніші підказки, коли гравець відчуває труднощі, або зменшувати кількість підказок по мірі зростання майстерності. Посилення також можуть пропонувати підказки або прямі рішення для конкретних клітинок.

Час зворотного зв'язку: Забезпечувати негайний зворотний зв'язок у реальному часі, щоб дозволити учням коригувати свої дії на місці та закріплювати знання, поки контекст ще свіжий.

Введення механік: Послідовно вводити нові механіки головоломок або операції, забезпечуючи, щоб гравець розумів та опановував їх, перш ніж збільшувати складність або поєднувати кілька механік.

Алгоритм адаптації (покроково):

Збір даних: Впровадити надійне відстеження даних про продуктивність гравця в реальному часі.

Проектування алгоритму: Розробити динамічні алгоритми (наприклад, зважені середні, прогностичні моделі), які коригують складність на основі зібраних даних. Найкраща адаптація є "тонкою, але ефективною", майже невидимою для гравця, як "легкий поштовх".

Реалізація: Бездоганно інтегрувати адаптивний алгоритм у цикл генерації головоломок та ігрового процесу.

Тестування та ітерація: Постійно тестувати та вдосконалювати адаптивну систему на основі відгуків гравців та спостережуваних поведінкових патернів. Поступовий підхід мінімізує ризик.

Виклики та етичні міркування: Адаптивна складність не позбавлена небезпек; надмірна корекція може бути катастрофічною, а прогностичні моделі можуть давати збої. Раптове підвищення складності після успіхів може зруйнувати довіру. Система завжди повинна відчуватися справедливою, ніколи не караючою, забезпечуючи, щоб гравець відчував контроль та мав свободу дій.

Хоча адаптивна складність високо оцінюється за підвищення інклюзивності та залученості , дослідження також прямо попереджають про небезпеки "надмірної корекції" та відчуття "караючої" системи. Це підкреслює критичне етичне міркування: як розробити алгоритм, щоб він справді підтримував навчання та сприяв майстерності , а не просто маніпулював залученістю або маскував відсутність справжнього прогресу. "Тонкий зсув" є

ключовим, що означає, що адаптація повинна відчуватися як підтримуючий посібник, а не контролююча або демотивуюча сила. Це призводить до висновку, що відстежувані метрики (наприклад, точність, час, використання підказок) повинні бути ретельно відібрані, щоб відображати прогрес у навчанні та когнітивне навантаження, а не лише поверхневу залученість. Алгоритм повинен бути прозорим у своїх намірах (навіть якщо він тонкий у виконанні) і потенційно пропонувати гравцям варіанти налаштування для збереження їхньої свободи дій, запобігаючи перетворенню системи на "чорну скриньку", яка диктує досвід без участі користувача. Це особливо важливо для освітньої гри, де кінцевою метою є розширення можливостей гравця та справжнє набуття навичок, а не просто розвага.

3 РЕАЛІЗАЦІЯ НАВЧАЛЬНОЇ КОМП'ЮТЕРНОЇ ГРИ-ГОЛОВОЛОМКИ «МАТЕМАТИЧНІ КРОСВОРДИ»

3.1 Добір засобів розробки навчальної комп'ютерної гри-головоломки «Математичні кросворди»

З метою забезпечення гнучкого процесу розробки, високої продуктивності та можливості адаптації гри до різних платформ, для реалізації комп'ютерної гри «Математичні головоломки» було вибрано бібліотеку React.

React — це відкрита JavaScript бібліотека, розроблена командою компанії Facebook, яка призначена для створення інтерфейсів користувача. Вона забезпечує ефективне і просте управління компонентами вебдодатків, дозволяючи розробникам будувати масштабовані, інтерактивні та швидкі вебінтерфейси. Основною метою React є вирішення проблем часткового оновлення вмісту вебсторінки, що значно підвищує продуктивність та зручність розробки.

React використовує концепцію компонентів — невеликих, незалежних і багаторазових елементів інтерфейсу, які можна легко комбінувати для створення складних застосунків. Однією з ключових особливостей бібліотеки є використання віртуального DOM (Virtual DOM), який скорочує кількість операцій оновлення реального DOM, оптимізуючи продуктивність вебдодатка. Завдяки цьому React здатний швидко та ефективно реагувати на зміни даних, миттєво оновлюючи інтерфейс користувача.

Ще однією важливою особливістю React є декларативний підхід до програмування, який дає змогу розробникам концентруватися на тому, що саме потрібно відображати, а не на тому, як це зробити. Це полегшує навчання та роботу з бібліотекою, роблячи код більш читабельним і зрозумілим.

React також підтримує концепцію управління станом через бібліотеки на зразок Redux або Context API, що спрощує управління даними в додатку. Крім того, React може бути використаний не лише для створення класичних

вебдодатків, а й для розробки мобільних (за допомогою React Native), десктопних і навіть серверних рішень.

Важливо зазначити, що React є відкритим продуктом із великою спільнотою розробників. Це означає, що існує величезна кількість доповнень, бібліотек і готових компонентів, які дозволяють розширювати можливості React і прискорювати процес розробки. Завдяки цьому він став одним із найпопулярніших інструментів для створення сучасних вебінтерфейсів.

Для зберігання даних про гравця та його досягнення ми скористались Firebase – хмарною платформою для розробки мобільних та веб-додатків, яка забезпечує високий рівень продуктивності, безпеки та масштабованості. Основним інструментом для управління цією інформацією стала база даних Firestore, що є частиною Firebase. Firestore дозволяє зберігати дані у вигляді документів, організованих у колекції, що робить структуру даних гнучкою та ефективною для різних сценаріїв використання.

У нашому випадку для кожного гравця створюється окрема колекція або документ, де записуються його особисті дані, такі як ім'я, унікальний ідентифікаційний номер чи електронна пошта. У цих же документах можна зберігати інформацію про досягнення, включаючи статистику, рівні, нагороди або інші важливі дані, які відображають прогрес гравця.

Firestore забезпечує реальний час синхронізації даних, що означає, що будь-які зміни, внесені до профілю гравця або його досягнень, миттєво оновлюються у додатку без необхідності перезавантаження. Крім того, у базі даних доступні функції швидкого пошуку і запитів, що спрощує отримання потрібної інформації з великого обсягу даних.

Безпечність даних є ще однією ключовою особливістю Firestore. Використовуючи правила доступу та аутентифікації, можна обмежити доступ до інформації, дозволяючи лише авторизованим користувачам переглядати або змінювати свої дані. Це забезпечує конфіденційність та захищеність інформації.

Завдяки таким можливостям Firebase та Firestore значно спрощуються процеси управління даними гравців, їх аналізу та інтеграції з додатком, а також забезпечується стабільна робота системи навіть за умов високого рівня навантаження.

3.2 Огляд результатів розробки навчальної комп'ютерної гри-головоломки «Математичні кросворди»

Вся функціональність нашої гри, що передбачає декілька головоломок (Математичний кросворд, КенКен, Какуро) інтегрована в один React-застосунок.

Стартова сторінка показана на рисунку 3.1.

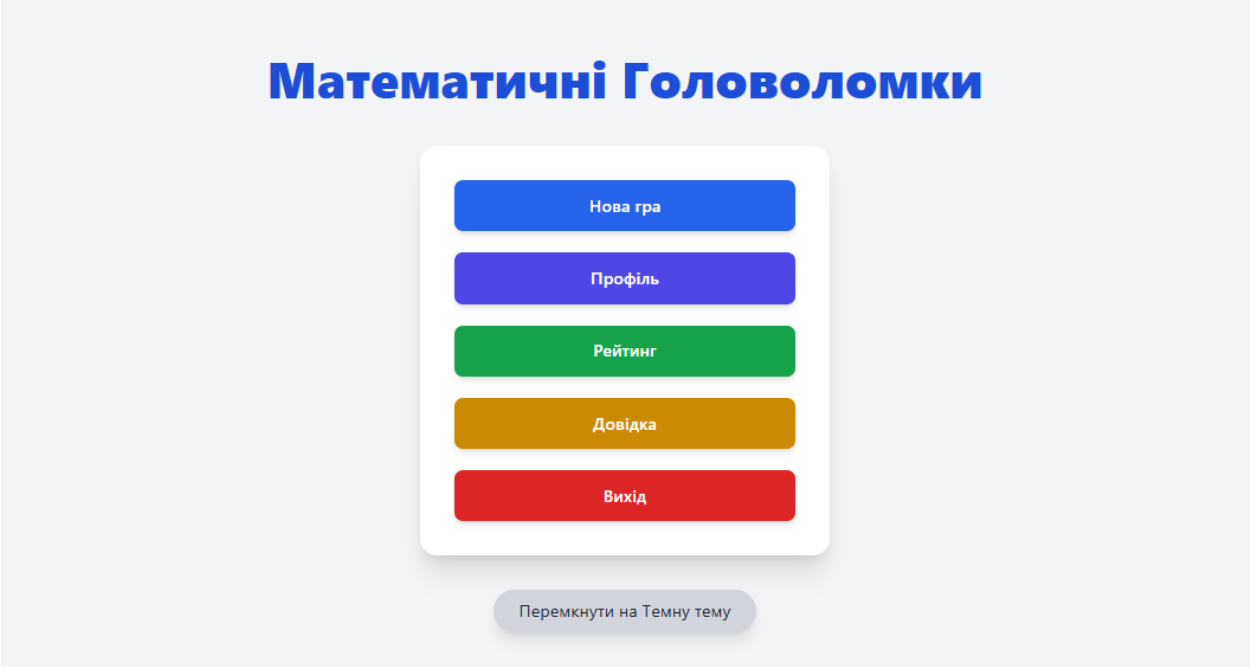

Рисунок 3.1 – Стартова сторінка навчальної гри

На стартовій сторінці відображається головне меню з кнопками:

- "Нова гра": Відкриває підменю для вибору однієї з трьох головоломок.
- "Профіль": Веде до підменю, де ви можете вибрати, профіль якої гри переглянути (ваші найкращі часи).

- "Рейтинг": Веде до підменю, де ви можете вибрати, рейтинг якої гри переглянути (глобальні найкращі часи).
- "Довідка": Веде до підменю, де ви можете вибрати, довідку по якій грі переглянути.
- "Вихід": Повертає на головну сторінку застосунку.

При виборі Нової гри користувач потрапляє в меню з вибором типу головолоски та рівня складності (рис. 3.2).

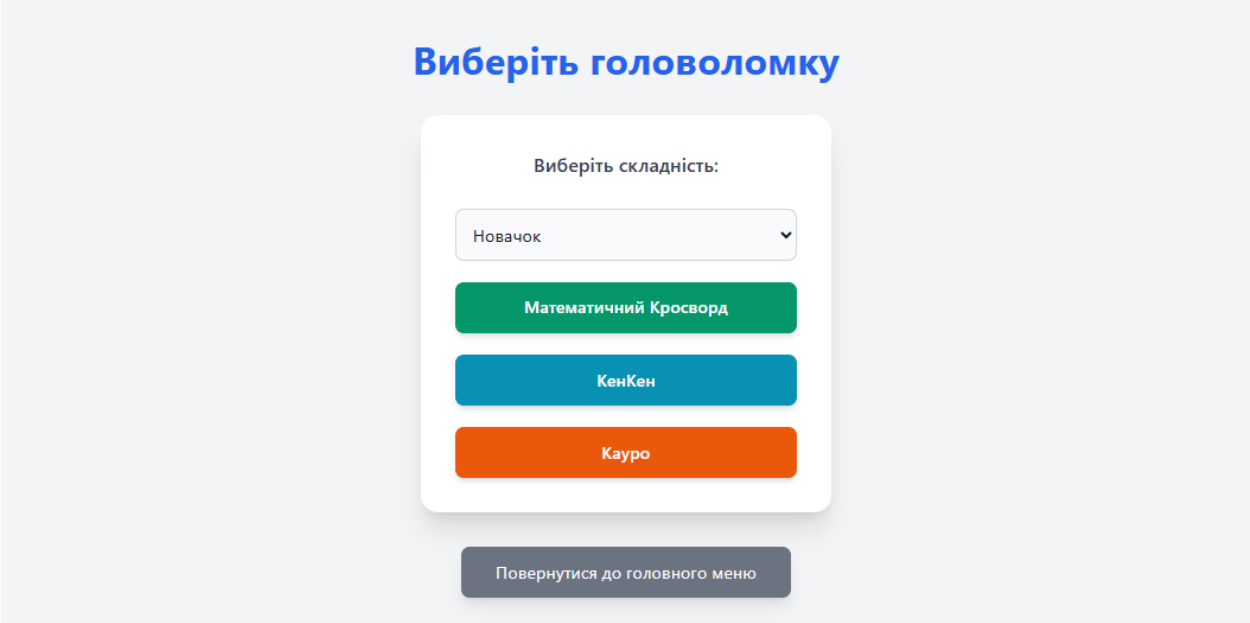

Рисунок 3.2 –Вибір головоломки

В подальшому в меню можна додавати і інші головоломки, але на даний момент реалізовано три:

- Математичний кросворд
- КенКен
- Какуро

Також можна обрати один з рівнів складності:

- Новачок
- Досвідчений
- Професіонал
- Експерт

- Геній

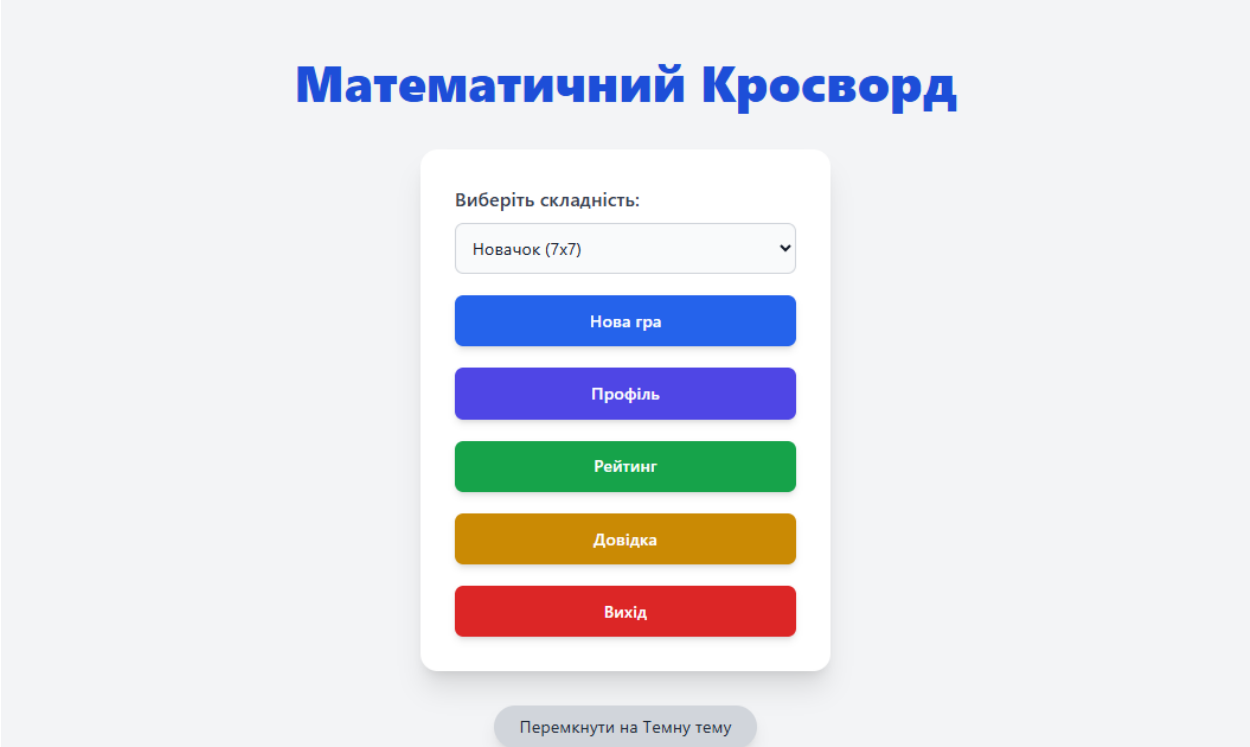

Рисунок 3.3 – Початок гри Математичний кросворд

На рисунку 3.4 показано поле математичного кросворду.

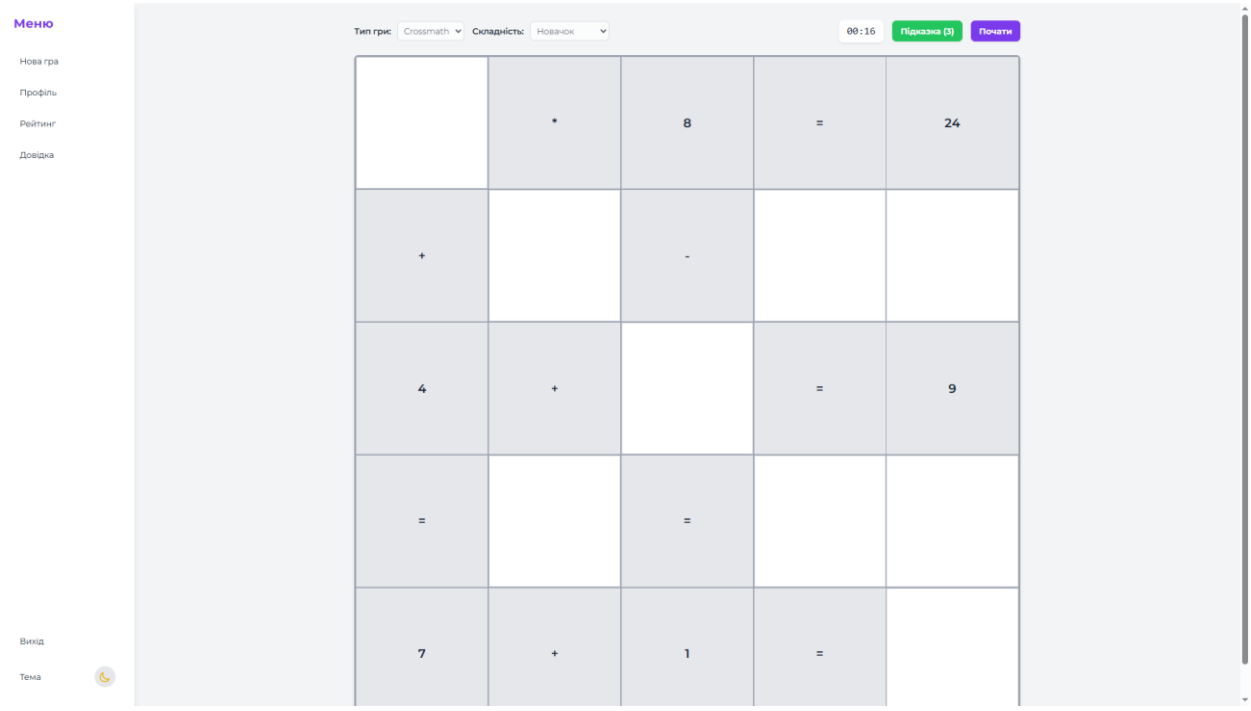

Рисунок 3.4 – Математичний кросворд

Інша головоломка – КенКен показана на рисунку 3.5.

Головоломки генеруються на основі латинських квадратів та випадково створених "камер" з арифметичними операціями та цільовими значеннями. Алгоритм забезпечує, що операції (додавання, множення, віднімання, ділення) застосовуються коректно.

Користувачі можуть вводити числа в клітинки сітки. Застосунок візуально підказує, якщо введене число не відповідає правильному розв'язанню.

Рівні складності: Реалізовано 5 рівнів складності: "Новачок" (4x4), "Досвідчений" (5x5), "Професіонал" (6x6), "Експерт" (7x7), "Геній" (8x8). Розмір сітки змінюється залежно від обраного рівня.

В кожній грі таймер відстежує час, витрачений на розв'язання головоломки, і зупиняється після її завершення.

Кнопка "Підказки" дозволяє отримати до 3 підказок за гру, які розкривають правильне число у випадковій порожній або неправильно заповненій клітинці.

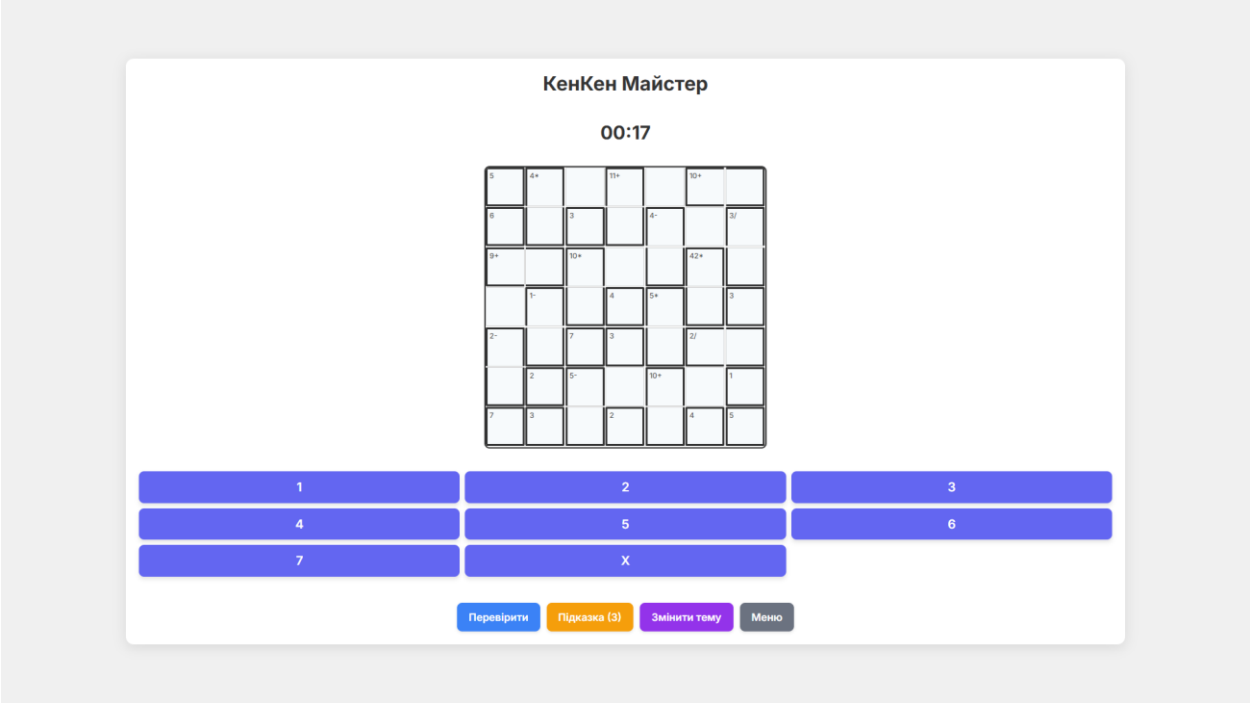

Рисунок 3.5 – Гра КенКен

Кнопка "Перемкнути тему" дозволяє перемикатися між світлою та темною темами інтерфейсу (рис. 3.6).

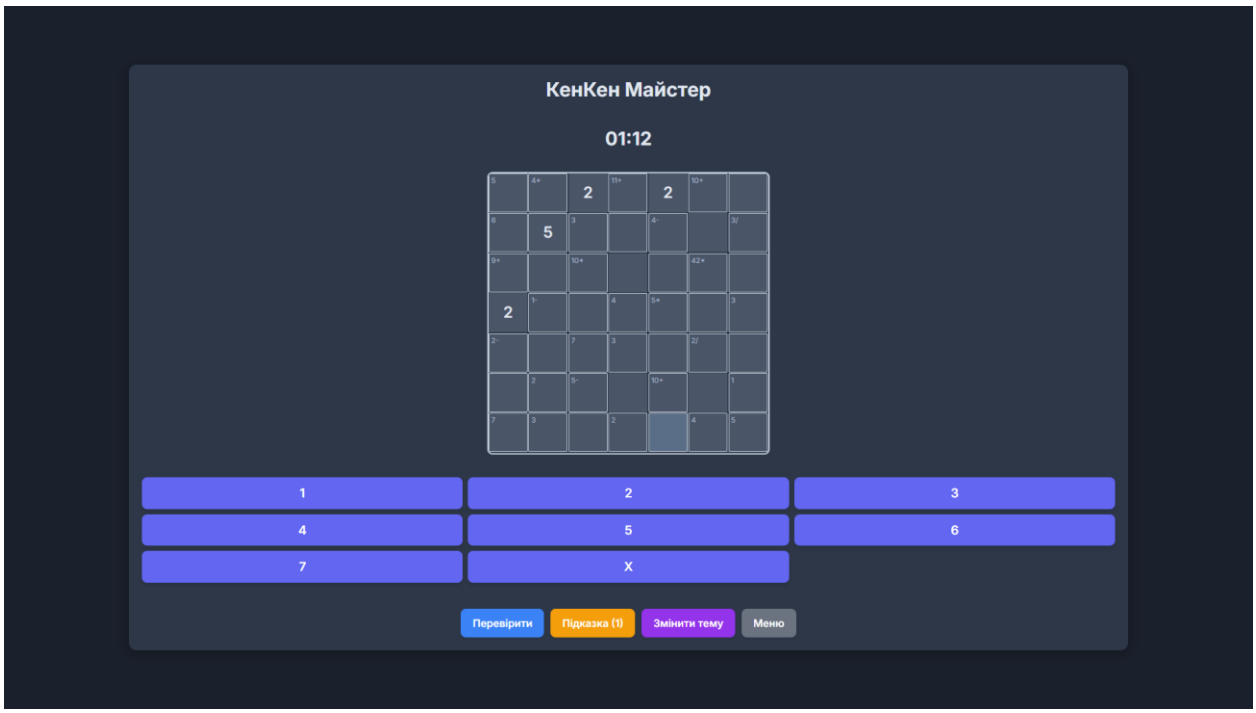

Рисунок 3.6 – Темна тема

При перевірці головоломки можуть виводитись повідомлення про помилку в розв’язанні (рис. 3.7), неповне розв’язання (рис. 3.8) або що головоломка успішно розв’язана (рис. 3.9)

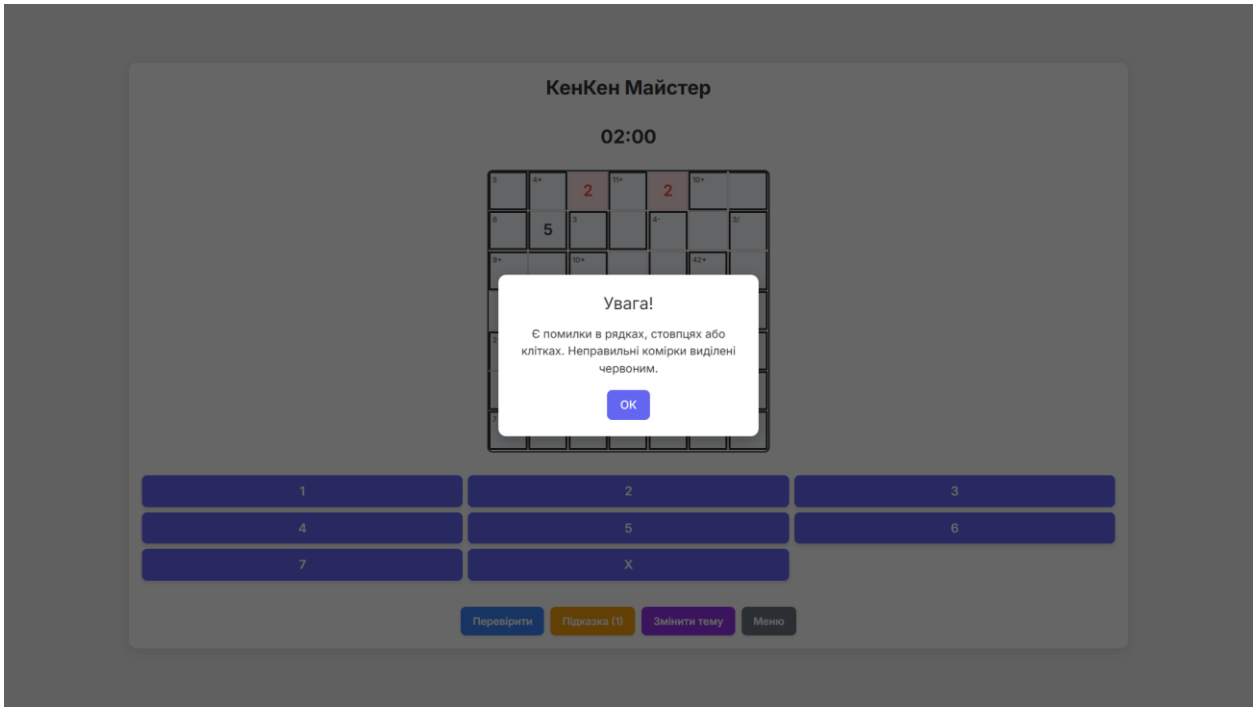

Рисунок 3.7 – Повідомлення про помилку під час розв’язання головоломки

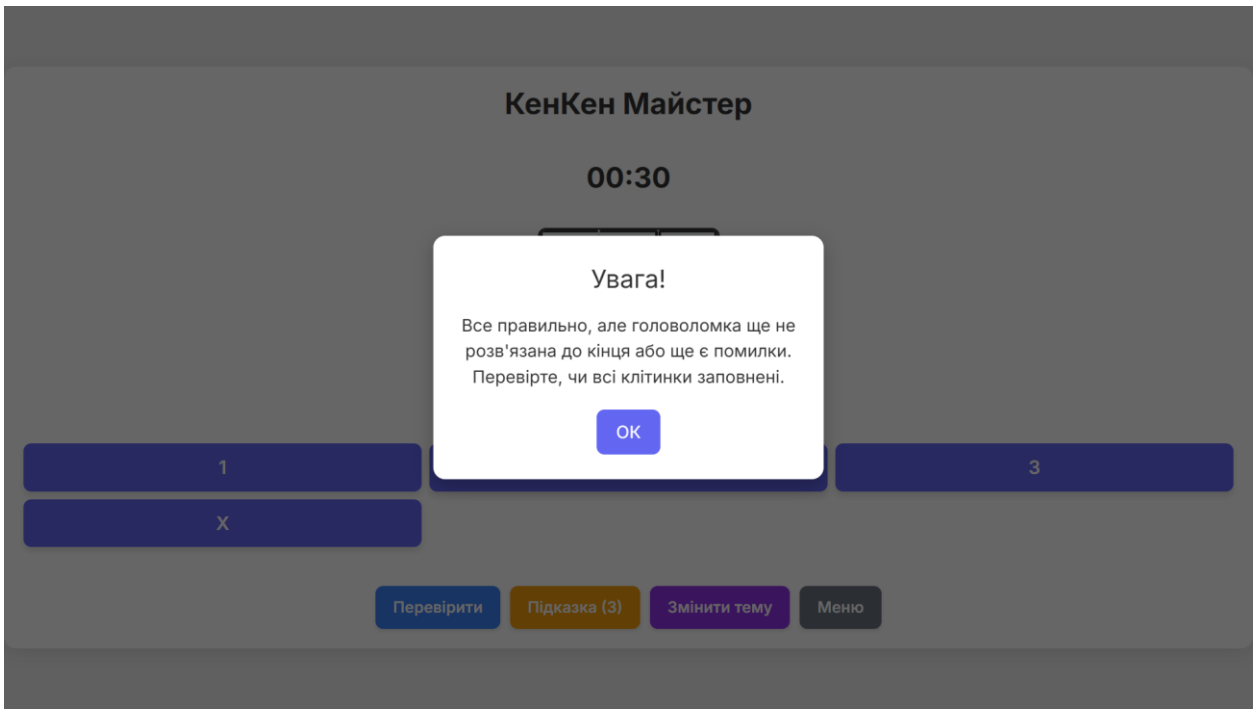

Рисунок 3.8 – Неповне розв’язання головоломки

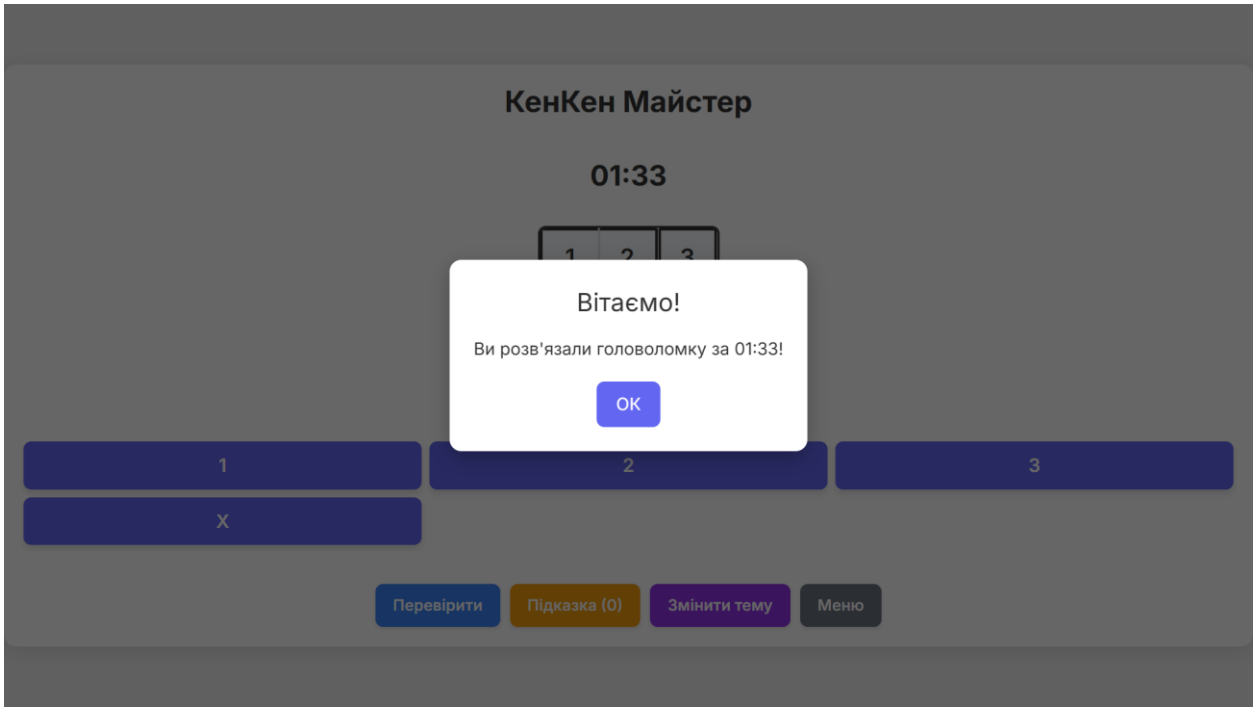

Рисунок 3.9 – Головоломка повністю ров’язана

Головне меню також містить пункти "Нова гра", "Профіль", "Рейтинг", "Довідка" та "Вихід".

"Профіль": Відображає найкращі часи користувача для кожного рівня складності (рис. 3.10).

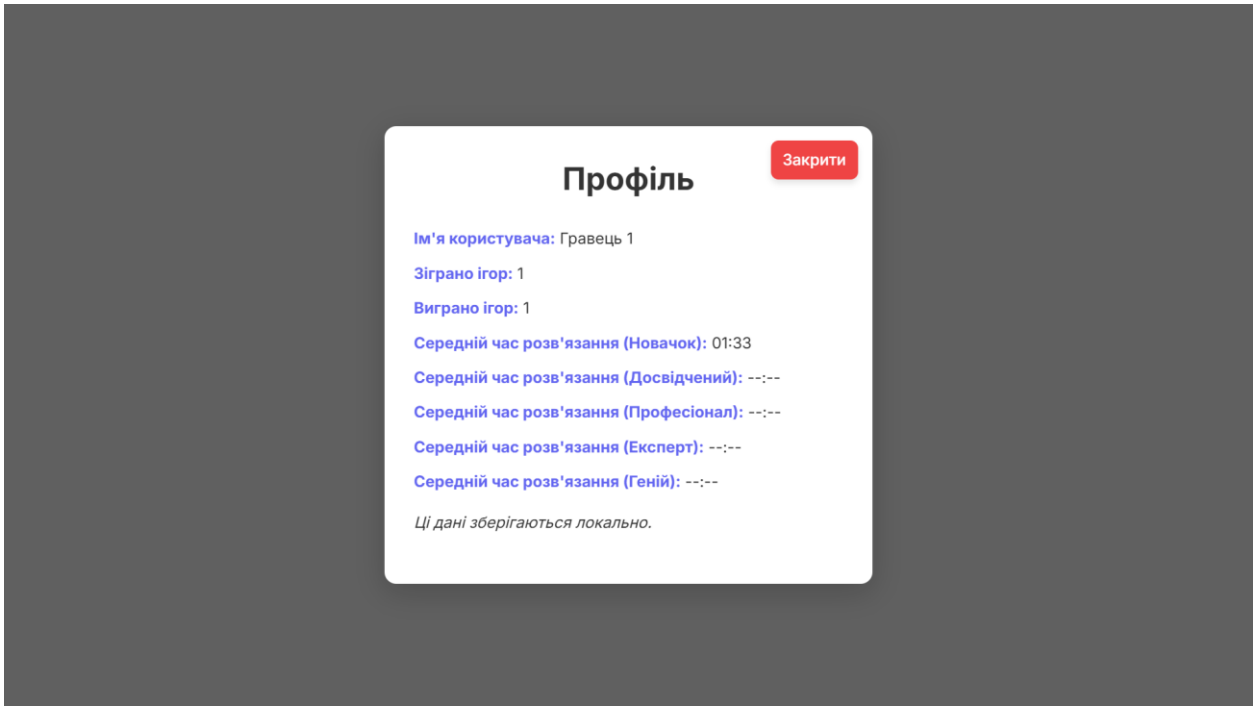

Рисунок 3.10 – Профіль гравця

"Рейтинг": Показує глобальний рейтинг найкращих часів для всіх користувачів (рис. 3.12).

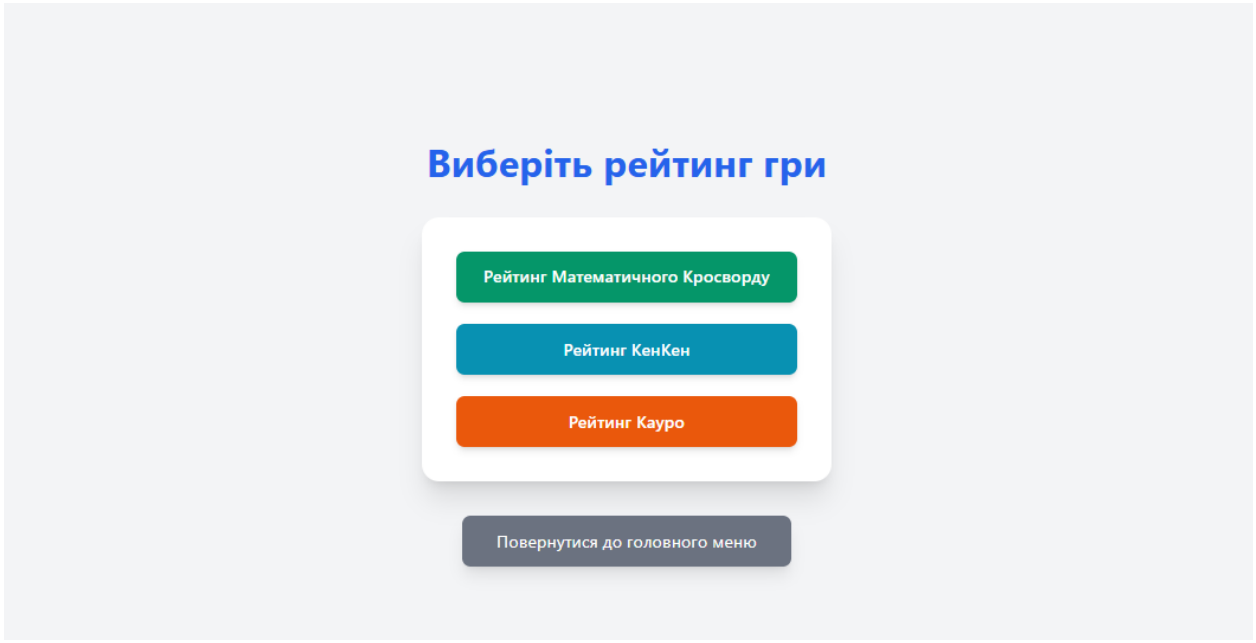

Рисунок 3.11 – Вибір гри для перегляду рейтингу

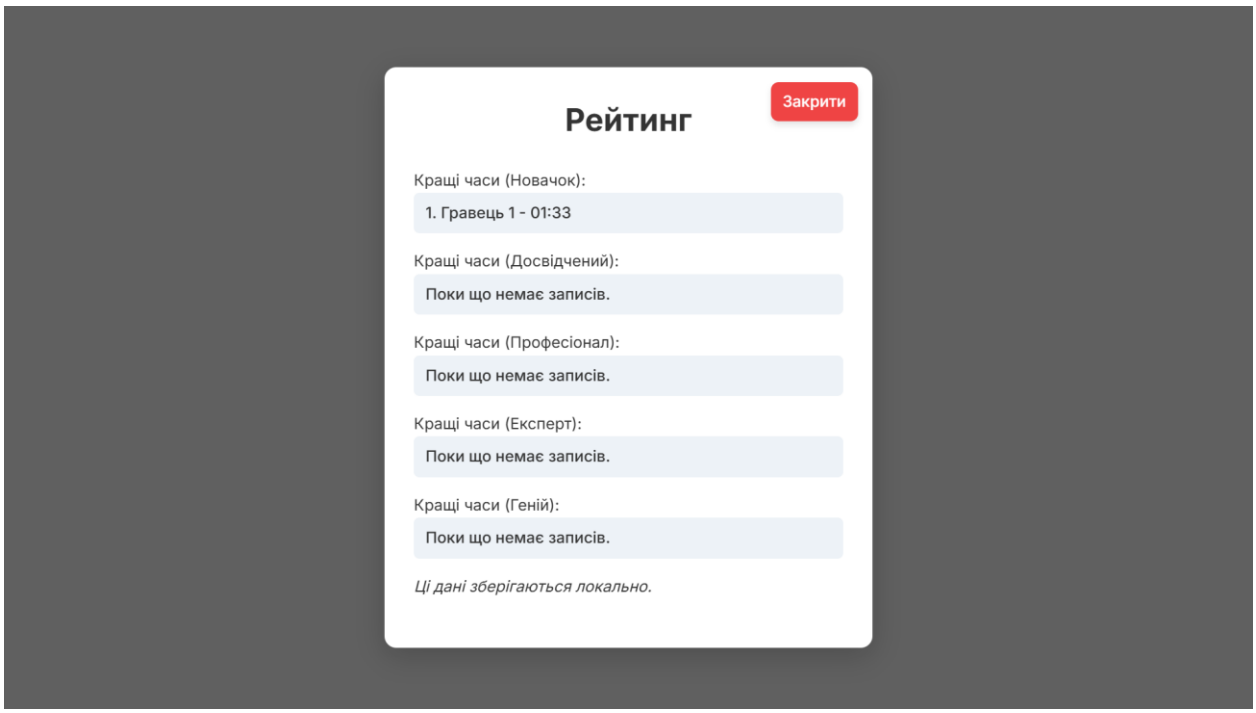

Рисунок 3.12 – Рейтинг

"Довідка": Надає детальні правила гри в КенКен (рис. 3.13).

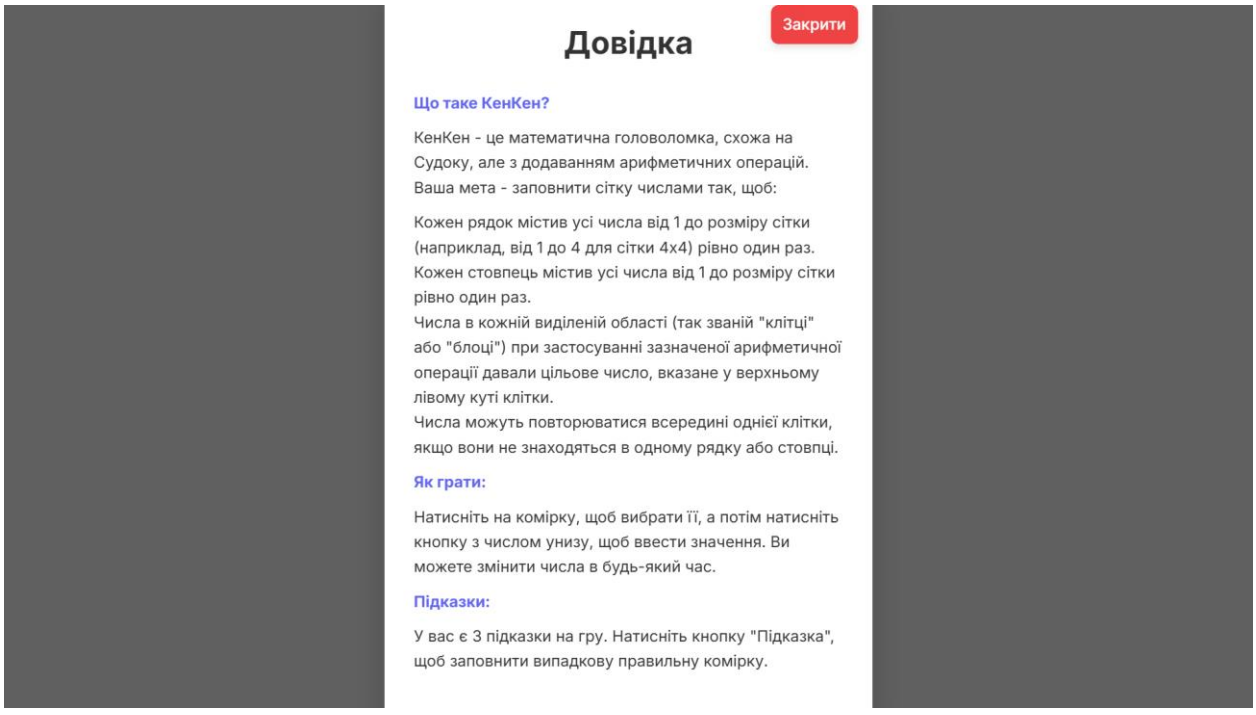

Рисунок 3.13 – Довідка по правилам гри

"Вихід": Повертає до головного меню.

Головоломка Какуро має подібний інтерфейс (рис. 3.14).

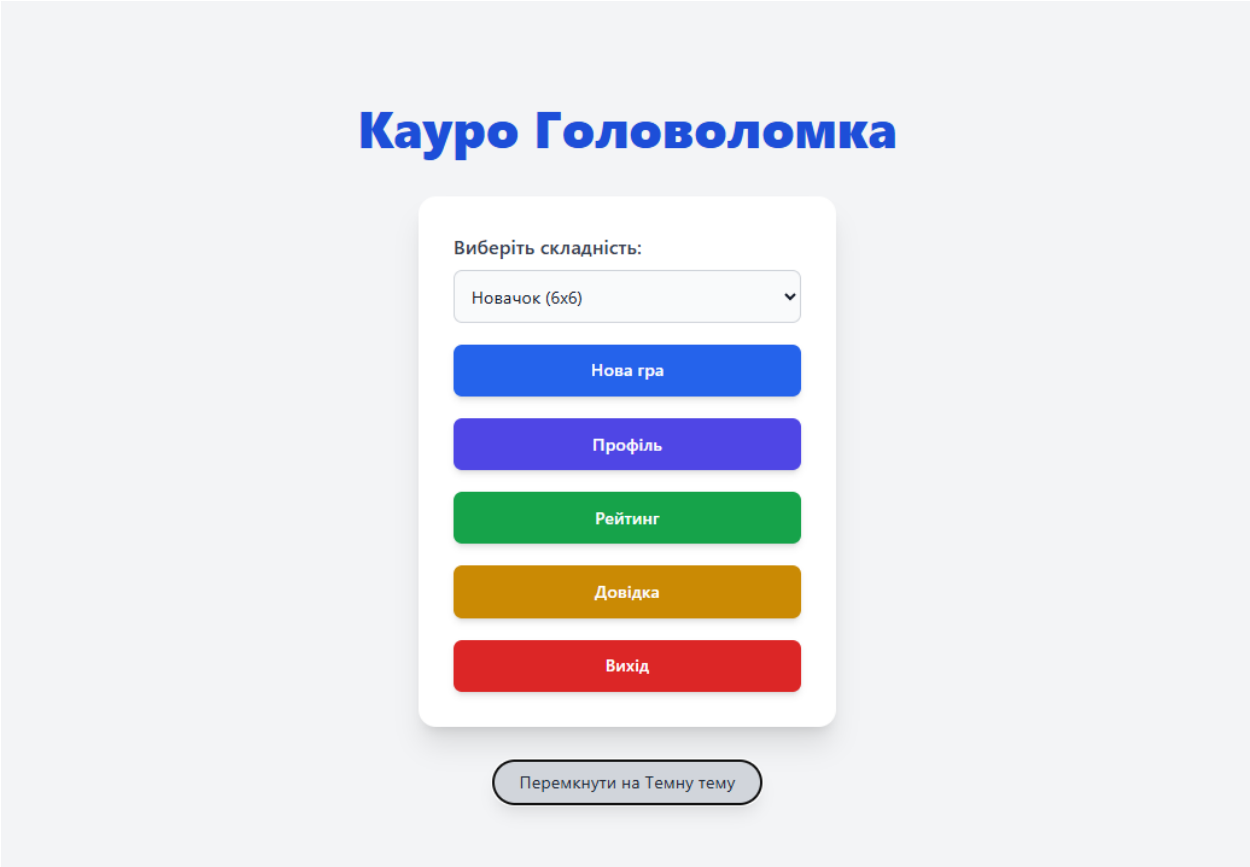

Рисунок 3.14 – Меню головоломки Какуро

Застосунок створює сітку Кауро, розміщує чорні клітинки та генерує горизонтальні й вертикальні "пробіги" з унікальними числами та відповідними сумами. Користувачі можуть вводити числа (від 1 до 9) у білі клітинки сітки. Неправильно введені числа виділяються червоним.

Рівні складності: Реалізовано 5 рівнів складності: "Новачок" (6x6), "Досвідчений" (7x7), "Професіонал" (8x8), "Експерт" (9x9), "Геній" (10x10). Розмір сітки змінюється залежно від обраного рівня, а також змінюється щільність чорних клітинок.

Таймер відстежує час, витрачений на розв'язання головоломки, і зупиняється після її завершення.

Кнопка "Підказки" дозволяє отримати до 3 підказок за гру, які розкривають правильне число у випадковій порожній або неправильно заповненій клітинці.

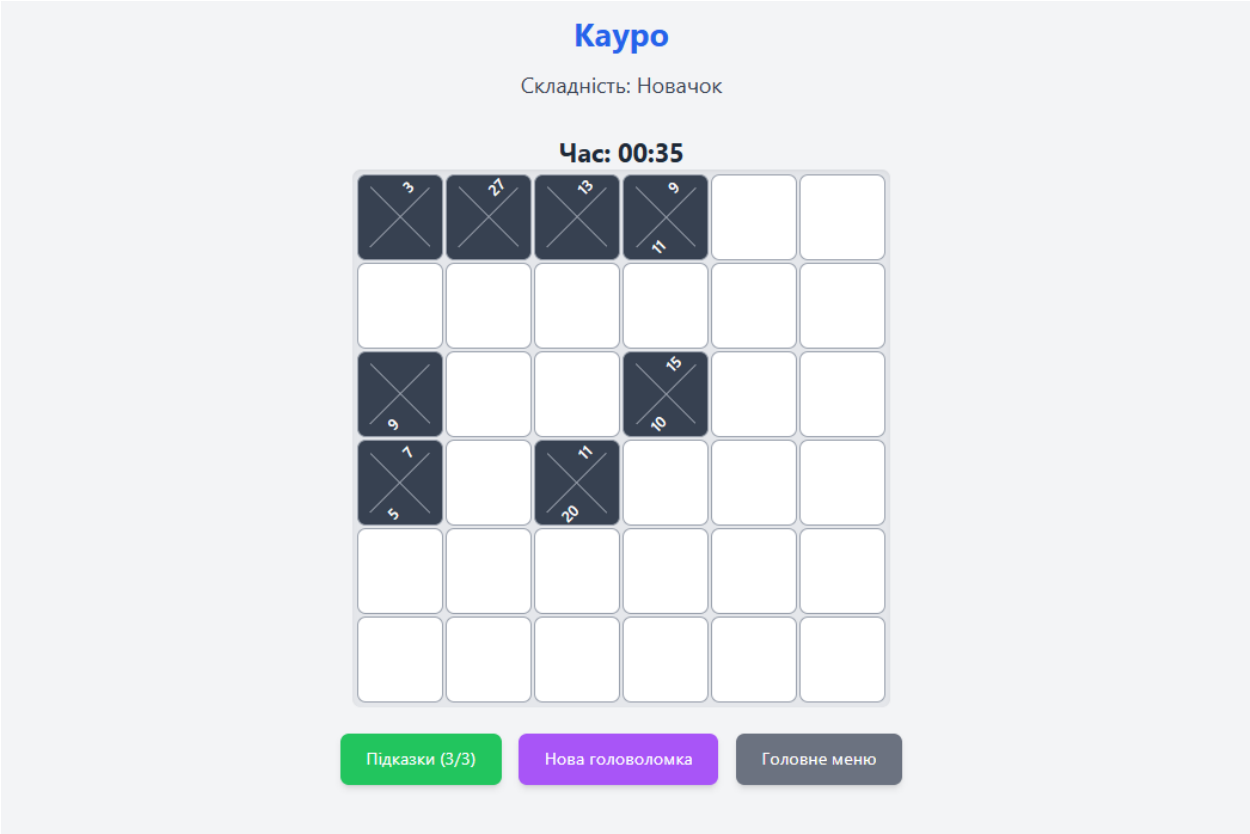

Рисунок 3.15 – Головоломка Какуро

Для Какуро також відображається довідка (рис. 3.16).

Кнопка "Перемкнути тему" дозволяє перемикатися між світлою та темною темами інтерфейсу.

Інтерфейс кожної головоломки адаптований для коректного відображення на різних розмірах екранів, в тому числі на мобільних пристроях.

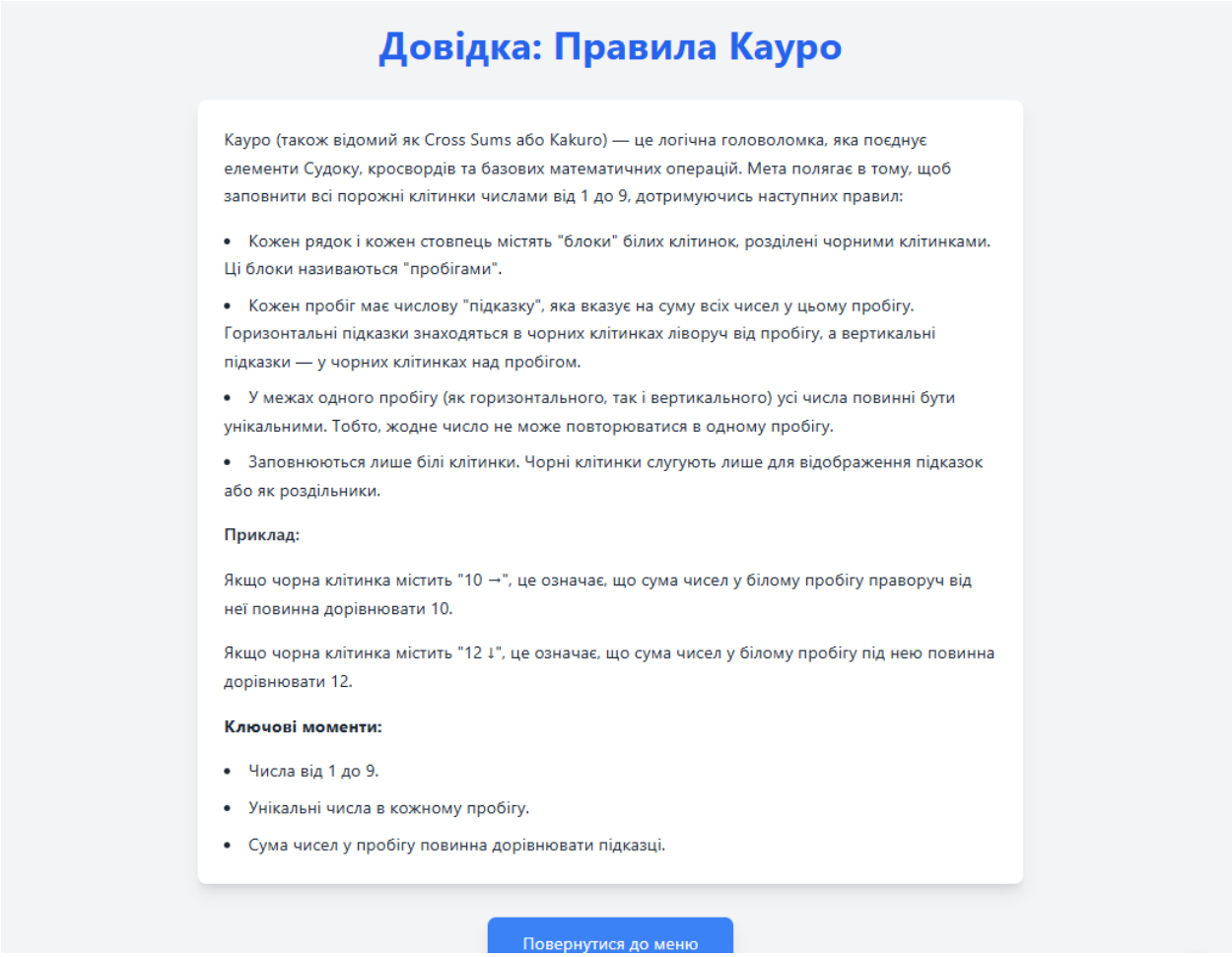

Рисунок 3.16 – Правила гри Какуро

Таким чином ми створили застосунок, що реалізує найцікавіші та найпопулярніші математичні головоломки. Створена гра може як розважити, так і бути засобом навчання та розвитку математичних навичок.

ВИСНОВКИ

Розробка навчальної комп'ютерної гри-головоломки "Математичні Кросворди" вимагає комплексного підходу, який поєднує глибоке розуміння математичних головоломок, педагогічних принципів та сучасних технічних вимог. Ключовим для успіху буде синтез механік Какуро та КенКен, що дозволить створити динамічні та різноманітні математичні завдання, які виходять за рамки простого додавання.

Освітня цінність гри буде значно підвищена завдяки впровадженню адаптивної складності, яка динамічно підлаштовується під рівень майстерності гравця, забезпечуючи постійний виклик без розчарування. Ця адаптація, у поєднанні з "майже нескінченною" генерацією головоломок, гарантуватиме, що гра залишається свіжою та ефективною як інструмент для розвитку арифметичної вправності, логічної дедукції та стійкості до розв'язання проблем.

З технічної точки зору, важливо створювати продукти, які будуть працювати на різних платформах, щоб охопити більше користувачів і зробити роботу ефективнішою. Для цього потрібно обрати зручний та надійний інструмент, який дозволить реалізувати складні головоломки і зробити простий і зрозумілий дизайн для користувачів. Важливо також добре налаштувати продуктивність і систему підтримки, щоб усе працювало без збоїв і людям було комфортно користуватися продуктом. Це допоможе залучати і утримувати більше людей.

Підсумовуючи, "Математичні Кросворди" мають потенціал стати потужним освітнім інструментом, який робить вивчення математики захопливим та доступним. Успіх залежатиме від бездоганної інтеграції інноваційних ігрових механік, педагогічно обґрунтованих систем прогресу та надійної, зручної для користувача технічної реалізації.

ПЕРЕЛІК ПОСИЛАНЬ

1. Math Blaster! – URL: https://en.wikipedia.org/wiki/Math_Blaster!
2. Number Munchers – URL: https://en.wikipedia.org/wiki/Number_Munchers
3. Basic Math – URL: https://www.mobygames.com/game/8843/basic-math/
4. Prodigy Math Game – URL: https://en.wikipedia.org/wiki/Prodigy_Math_Game
5. HyperRogue – URL: https://store.steampowered.com/app/342610/HyperRogue
6. DragonBox: Numbers – URL: https://dragonbox.com/products/numbers
7. Crossmath - Math Puzzle Games – URL: https://play.google.com/store/apps/details?id=math.puzzle.games.crossmath.number.puzzles.free
8. Kakuro – URL: https://en.wikipedia.org/wiki/Kakuro
9. Kakuro: Number Crossword – URL: https://play.google.com/store/apps/details?id=com.conceptispuzzles.kakuro
10. KenKen – URL: https://play.google.com/store/apps/details?id=kenkenclassic.com
11. Math Crossword — Number puzzle – URL: https://play.google.com/store/apps/details?id=com.zm.crossmath

Додаток А

Текст програми

```
    import React, { useState, useEffect, useRef, useCallback }
from 'react';
    import { initializeApp } from 'firebase/app';
    import { getAuth, signInAnonymously, signInWithCustomToken,
onAuthStateChanged } from 'firebase/auth';
    import { getFirestore, doc, getDoc, setDoc, onSnapshot,
collection, query, addDoc, getDocs, updateDoc } from
'firebase/firestore';

    // Define Firebase globals if not already defined (for Canvas
environment)
    const firebaseConfig = typeof __firebase_config !==
'undefined' ? JSON.parse(__firebase_config) : null;
    const initialAuthToken = typeof __initial_auth_token !==
'undefined' ? __initial_auth_token : null;
    const appId = typeof __app_id !== 'undefined' ? __app_id :
'default-math-puzzles-hub-app-id';

    // Initialize Firebase outside of components to avoid re-
initialization
    let firebaseApp, db, auth;
    if (firebaseConfig) {
      firebaseApp = initializeApp(firebaseConfig);
      db = getFirestore(firebaseApp);
      auth = getAuth(firebaseApp);
    }

    // --- Shared Components (Timer, Modal) ---
    const Timer = ({ time }) => {
      const formatTime = (seconds) => {
        const minutes = Math.floor(seconds / 60);
        const remainingSeconds = seconds % 60;
        return `${minutes.toString().padStart(2,
'0')}:${remainingSeconds.toString().padStart(2, '0')}`;
      };
      return (
        <div className="text-2xl font-bold text-gray-800
dark:text-gray-200 mt-4">
          Час: {formatTime(time)}
        </div>
      );
    };

    const Modal = ({ isOpen, onClose, children }) => {
      if (!isOpen) return null;
      return (
        <div className="fixed inset-0 bg-gray-600 bg-opacity-50
flex items-center justify-center z-50">
```

```
          <div className="bg-white dark:bg-gray-800 p-6 rounded-
lg shadow-xl max-w-sm w-full text-center">
            {children}
            <button
              onClick={onClose}
              className="mt-4 px-6 py-2 bg-blue-500 hover:bg-
blue-600 text-white rounded-lg transition-colors"
            >
              Закрити
            </button>
          </div>
        </div>
      );
    };

    // --- Mathecross Components (Copied from previous version)
---
    // Helper function to generate a valid mathematical expression
(num1 op num2 = result)
    const generateValidMathecrossExpression = (n1_initial,
op_initial, n2_initial, res_initial) => {
      const operators = ['+', '-', '*', '/'];
      const numbers = Array.from({ length: 9 }, (_, i) => i + 1);

      const calculate = (n1_val, op_val, n2_val) => {
        switch (op_val) {
          case '+': return n1_val + n2_val;
          case '-': return n1_val - n2_val;
          case '*': return n1_val * n2_val;
          case '/': return (n2_val !== 0 && n1_val % n2_val ===
0) ? n1_val / n2_val : NaN;
          default: return NaN;
        }
      };

      const possibleN1 = n1_initial !== null ? [n1_initial] :
numbers;
      const possibleOp = op_initial !== null ? [op_initial] :
operators;
      const possibleN2 = n2_initial !== null ? [n2_initial] :
numbers;

      for (const n1 of possibleN1) {
        for (const op of possibleOp) {
          for (const n2 of possibleN2) {
            const calculated_res = calculate(n1, op, n2);
            if (
              !isNaN(calculated_res) &&
              calculated_res >= 1 && calculated_res <= 9 &&
              Number.isInteger(calculated_res) &&
              (res_initial === null || calculated_res ===
res_initial)
            ) {
```

```
                return { num1: n1, op: op, num2: n2, result:
calculated_res, success: true };
              }
            }
          }
        }
        return { success: false };
      };

      const mathecrossPuzzleTemplates = {
        'Новачок': {
          size: 7,
          layout: [
            ['b', 'b', 'b', 'b', 'b', 'b', 'b'],
            ['b', 'n', 'o', 'n', 'e', 'n', 'b'],
            ['b', 'b', 'b', 'n', 'b', 'b', 'b'],
            ['b', 'b', 'b', 'o', 'b', 'b', 'b'],
            ['b', 'b', 'b', 'n', 'b', 'b', 'b'],
            ['b', 'b', 'b', 'e', 'b', 'b', 'b'],
            ['b', 'b', 'b', 'n', 'b', 'b', 'b'],
          ],
          equations: [
            { id: 'H1', cells: [[1,1], [1,2], [1,3], [1,4], [1,5]]
},
            { id: 'V1', cells: [[1,3], [2,3], [3,3], [4,3], [5,3]]
}
          ],
          blanksRatio: 0.5
        },
        'Досвідчений': {
          size: 9,
          layout: [
            ['b', 'b', 'b', 'b', 'b', 'b', 'b', 'b', 'b'],
            ['b', 'n', 'o', 'n', 'e', 'n', 'b', 'b', 'b'],
            ['b', 'b', 'b', 'n', 'b', 'b', 'b', 'b', 'b'],
            ['b', 'n', 'o', 'n', 'e', 'n', 'b', 'b', 'b'],
            ['b', 'b', 'b', 'n', 'b', 'b', 'b', 'b', 'b'],
            ['b', 'b', 'b', 'n', 'b', 'b', 'b', 'b', 'b'],
            ['b', 'b', 'b', 'o', 'b', 'b', 'b', 'b', 'b'],
            ['b', 'b', 'b', 'n', 'b', 'b', 'b', 'b', 'b'],
            ['b', 'b', 'b', 'e', 'b', 'b', 'b', 'b', 'b'],
            ['b', 'b', 'b', 'n', 'b', 'b', 'b', 'b', 'b'],
          ],
          equations: [
            { id: 'H1', cells: [[1,1], [1,2], [1,3], [1,4], [1,5]]
},
            { id: 'H2', cells: [[3,1], [3,2], [3,3], [3,4], [3,5]]
},
            { id: 'V1', cells: [[1,3], [2,3], [3,3], [4,3], [5,3]]
},
            { id: 'V2', cells: [[5,3], [6,3], [7,3], [8,3], [9,3]]
}
          ],
```

```
         blanksRatio: 0.6
       },
       'Професіонал': {
         size: 11,
         layout: [
           ['b', 'b', 'b', 'b', 'b', 'b', 'b', 'b', 'b', 'b', 'b'],
           ['b', 'n', 'o', 'n', 'e', 'n', 'b', 'n', 'o', 'n', 'e',
'n'],
           ['b', 'b', 'b', 'n', 'b', 'b', 'b', 'b', 'b', 'n', 'b'],
           ['b', 'n', 'o', 'n', 'e', 'n', 'b', 'n', 'o', 'n', 'e',
'n'],
           ['b', 'b', 'b', 'n', 'b', 'b', 'b', 'b', 'b', 'n', 'b'],
           ['b', 'n', 'o', 'n', 'e', 'n', 'b', 'n', 'o', 'n', 'e',
'n'],
           ['b', 'b', 'b', 'n', 'b', 'b', 'b', 'b', 'b', 'n', 'b'],
           ['b', 'n', 'o', 'n', 'e', 'n', 'b', 'n', 'o', 'n', 'e',
'n'],
           ['b', 'b', 'b', 'n', 'b', 'b', 'b', 'b', 'b', 'n', 'b'],
           ['b', 'b', 'b', 'b', 'b', 'b', 'b', 'b', 'b', 'b', 'b'],
         ],
         equations: [
             { id: 'H1', cells: [[1,1], [1,2], [1,3], [1,4], [1,5]]
},
             { id: 'H2', cells: [[1,7], [1,8], [1,9], [1,10],
[1,11]] },
             { id: 'H3', cells: [[3,1], [3,2], [3,3], [3,4], [3,5]]
},
             { id: 'H4', cells: [[3,7], [3,8], [3,9], [3,10],
[3,11]] },
             { id: 'H5', cells: [[5,1], [5,2], [5,3], [5,4], [5,5]]
},
             { id: 'H6', cells: [[5,7], [5,8], [5,9], [5,10],
[5,11]] },
             { id: 'H7', cells: [[7,1], [7,2], [7,3], [7,4], [7,5]]
},
             { id: 'H8', cells: [[7,7], [7,8], [7,9], [7,10],
[7,11]] },

             { id: 'V1', cells: [[1,3], [2,3], [3,3], [4,3], [5,3]]
},
             { id: 'V2', cells: [[3,3], [4,3], [5,3], [6,3], [7,3]]
},
             { id: 'V3', cells: [[1,9], [2,9], [3,9], [4,9], [5,9]]
},
             { id: 'V4', cells: [[3,9], [4,9], [5,9], [6,9], [7,9]]
},
         ],
         blanksRatio: 0.7
       },
       'Експерт': {
         size: 13,
         layout: [
```

```
           ['b', 'b', 'b', 'b', 'b', 'b', 'b', 'b', 'b', 'b', 'b',
'b', 'b'],
           ['b', 'n', 'o', 'n', 'e', 'n', 'b', 'n', 'o', 'n', 'e',
'n', 'b'],
           ['b', 'b', 'b', 'n', 'b', 'b', 'b', 'b', 'b', 'n', 'b',
'b', 'b'],
           ['b', 'n', 'o', 'n', 'e', 'n', 'b', 'n', 'o', 'n', 'e',
'n', 'b'],
           ['b', 'b', 'b', 'n', 'b', 'b', 'b', 'b', 'b', 'n', 'b',
'b', 'b'],
           ['b', 'n', 'o', 'n', 'e', 'n', 'b', 'n', 'o', 'n', 'e',
'n', 'b'],
           ['b', 'b', 'b', 'n', 'b', 'b', 'b', 'b', 'b', 'n', 'b',
'b', 'b'],
           ['b', 'n', 'o', 'n', 'e', 'n', 'b', 'n', 'o', 'n', 'e',
'n', 'b'],
           ['b', 'b', 'b', 'n', 'b', 'b', 'b', 'b', 'b', 'n', 'b',
'b', 'b'],
           ['b', 'n', 'o', 'n', 'e', 'n', 'b', 'n', 'o', 'n', 'e',
'n', 'b'],
           ['b', 'b', 'b', 'n', 'b', 'b', 'b', 'b', 'b', 'n', 'b',
'b', 'b'],
           ['b', 'n', 'o', 'n', 'e', 'n', 'b', 'n', 'o', 'n', 'e',
'n', 'b'],
           ['b', 'b', 'b', 'b', 'b', 'b', 'b', 'b', 'b', 'b', 'b',
'b', 'b'],
         ],
         equations: [
             { id: 'H1', cells: [[1,1], [1,2], [1,3], [1,4], [1,5]]
},
             { id: 'H2', cells: [[1,7], [1,8], [1,9], [1,10],
[1,11]] },
             { id: 'H3', cells: [[3,1], [3,2], [3,3], [3,4], [3,5]]
},
             { id: 'H4', cells: [[3,7], [3,8], [3,9], [3,10],
[3,11]] },
             { id: 'H5', cells: [[5,1], [5,2], [5,3], [5,4], [5,5]]
},
             { id: 'H6', cells: [[5,7], [5,8], [5,9], [5,10],
[5,11]] },
             { id: 'H7', cells: [[7,1], [7,2], [7,3], [7,4], [7,5]]
},
             { id: 'H8', cells: [[7,7], [7,8], [7,9], [7,10],
[7,11]] },
             { id: 'H9', cells: [[9,1], [9,2], [9,3], [9,4], [9,5]]
},
             { id: 'H10', cells: [[9,7], [9,8], [9,9], [9,10],
[9,11]] },
             { id: 'H11', cells: [[11,1], [11,2], [11,3], [11,4],
[11,5]] },
             { id: 'H12', cells: [[11,7], [11,8], [11,9], [11,10],
[11,11]] },
```

```
            { id: 'V1', cells: [[1,3], [2,3], [3,3], [4,3], [5,3]]
},
            { id: 'V2', cells: [[3,3], [4,3], [5,3], [6,3], [7,3]]
},
            { id: 'V3', cells: [[5,3], [6,3], [7,3], [8,3], [9,3]]
},
            { id: 'V4', cells: [[7,3], [8,3], [9,3], [10,3],
[11,3]] },

            { id: 'V5', cells: [[1,9], [2,9], [3,9], [4,9], [5,9]]
},
            { id: 'V6', cells: [[3,9], [4,9], [5,9], [6,9], [7,9]]
},
            { id: 'V7', cells: [[5,9], [6,9], [7,9], [8,9], [9,9]]
},
            { id: 'V8', cells: [[7,9], [8,9], [9,9], [10,9],
[11,9]] },
         ],
         blanksRatio: 0.75
       },
       'Геній': {
         size: 15,
         layout: [
           ['b', 'b', 'b', 'b', 'b', 'b', 'b', 'b', 'b', 'b', 'b',
'b', 'b', 'b', 'b'],
           ['b', 'n', 'o', 'n', 'e', 'n', 'b', 'n', 'o', 'n', 'e',
'n', 'b', 'n', 'o'],
           ['b', 'b', 'b', 'n', 'b', 'b', 'b', 'b', 'b', 'n', 'b',
'b', 'b', 'n', 'b'],
           ['b', 'n', 'o', 'n', 'e', 'n', 'b', 'n', 'o', 'n', 'e',
'n', 'b', 'e', 'n'],
           ['b', 'b', 'b', 'n', 'b', 'b', 'b', 'b', 'b', 'n', 'b',
'b', 'b', 'n', 'b'],
           ['b', 'n', 'o', 'n', 'e', 'n', 'b', 'n', 'o', 'n', 'e',
'n', 'b', 'o', 'n'],
           ['b', 'b', 'b', 'n', 'b', 'b', 'b', 'b', 'b', 'n', 'b',
'b', 'b', 'n', 'b'],
           ['b', 'n', 'o', 'n', 'e', 'n', 'b', 'n', 'o', 'n', 'e',
'n', 'b', 'e', 'n'],
           ['b', 'b', 'b', 'n', 'b', 'b', 'b', 'b', 'b', 'n', 'b',
'b', 'b', 'n', 'b'],
           ['b', 'n', 'o', 'n', 'e', 'n', 'b', 'n', 'o', 'n', 'e',
'n', 'b', 'o', 'n'],
           ['b', 'n', 'o', 'n', 'e', 'n', 'b', 'n', 'o', 'n', 'e',
'n', 'b', 'e', 'n'],
           ['b', 'b', 'b', 'n', 'b', 'b', 'b', 'b', 'b', 'n', 'b',
'b', 'b', 'n', 'b'],
           ['b', 'n', 'o', 'n', 'e', 'n', 'b', 'n', 'o', 'n', 'e',
'n', 'b', 'o', 'n'],
           ['b', 'b', 'b', 'b', 'b', 'b', 'b', 'b', 'b', 'b', 'b',
'b', 'b', 'b', 'b'],
```

```
        ],
        equations: [
            { id: 'H1', cells: [[1,1], [1,2], [1,3], [1,4], [1,5]]
},
            { id: 'H2', cells: [[1,7], [1,8], [1,9], [1,10],
[1,11]] },
            { id: 'H4', cells: [[3,1], [3,2], [3,3], [3,4], [3,5]]
},
            { id: 'H5', cells: [[3,7], [3,8], [3,9], [3,10],
[3,11]] },
            { id: 'H7', cells: [[5,1], [5,2], [5,3], [5,4], [5,5]]
},
            { id: 'H8', cells: [[5,7], [5,8], [5,9], [5,10],
[5,11]] },
            { id: 'H10', cells: [[7,1], [7,2], [7,3], [7,4],
[7,5]] },
            { id: 'H11', cells: [[7,7], [7,8], [7,9], [7,10],
[7,11]] },
            { id: 'H13', cells: [[9,1], [9,2], [9,3], [9,4],
[9,5]] },
            { id: 'H14', cells: [[9,7], [9,8], [9,9], [9,10],
[9,11]] },
            { id: 'H16', cells: [[11,1], [11,2], [11,3], [11,4],
[11,5]] },
            { id: 'H17', cells: [[11,7], [11,8], [11,9], [11,10],
[11,11]] },
            { id: 'H19', cells: [[13,1], [13,2], [13,3], [13,4],
[13,5]] },
            { id: 'H20', cells: [[13,7], [13,8], [13,9], [13,10],
[13,11]] },

            { id: 'V1', cells: [[1,3], [2,3], [3,3], [4,3], [5,3]]
},
            { id: 'V2', cells: [[3,3], [4,3], [5,3], [6,3], [7,3]]
},
            { id: 'V3', cells: [[5,3], [6,3], [7,3], [8,3], [9,3]]
},
            { id: 'V4', cells: [[7,3], [8,3], [9,3], [10,3],
[11,3]] },
            { id: 'V5', cells: [[9,3], [10,3], [11,3], [12,3],
[13,3]] },

            { id: 'V6', cells: [[1,9], [2,9], [3,9], [4,9], [5,9]]
},
            { id: 'V7', cells: [[3,9], [4,9], [5,9], [6,9], [7,9]]
},
            { id: 'V8', cells: [[5,9], [6,9], [7,9], [8,9], [9,9]]
},
            { id: 'V9', cells: [[7,9], [8,9], [9,9], [10,9],
[11,9]] },
            { id: 'V10', cells: [[9,9], [10,9], [11,9], [12,9],
[13,9]] },
        ],
```

```
        blanksRatio: 0.8
      },
    };

    const createMathecrossPuzzle = (difficulty) => {
      const template = mathecrossPuzzleTemplates[difficulty];
      if (!template) {
        console.error("Template not found for difficulty:",
difficulty);
        return null;
      }
      const size = template.size;
      let solutionGrid;
      let userGrid;
      let attempts = 0;
      const maxAttempts = 5000;

      while (attempts < maxAttempts) {
        let success = true;
        solutionGrid = Array(size).fill(0).map(() =>
Array(size).fill(null));

        for (let r = 0; r < size; r++) {
          for (let c = 0; c < size; c++) {
            const layoutChar = template.layout[r][c];
            let type;
            if (layoutChar === 'b') {
              type = 'black';
            } else if (['+', '-', '*', '/'].includes(layoutChar))
{
              type = 'operator';
            } else if (layoutChar === '=') {
              type = 'equals';
            } else {
              type = 'number';
            }
            solutionGrid[r][c] = { value: layoutChar, type: type,
isHidden: false, r, c };
          }
        }

        for (const eqDef of template.equations) {
          const [n1_cell, op_cell, n2_cell, eq_cell, res_cell] =
eqDef.cells.map(([r, c]) => solutionGrid[r][c]);
          let n1_known = n1_cell.value !== 'n' ?
parseInt(n1_cell.value) : null;
          let op_known = op_cell.value !== 'o' ? op_cell.value :
null;
          let n2_known = n2_cell.value !== 'n' ?
parseInt(n2_cell.value) : null;
          let res_known = res_cell.value !== 'n' ?
parseInt(res_cell.value) : null;
```

```
          const { num1, op, num2, result, success: eqGenSuccess
}    =    generateValidMathecrossExpression(n1_known,    op_known,
n2_known, res_known);

          if (!eqGenSuccess) {
            success = false;
            break;
          }
          solutionGrid[n1_cell.r][n1_cell.c].value             =
num1.toString();
          solutionGrid[n1_cell.r][n1_cell.c].type = 'number';
          if           (op_cell.value           ===           'o')
solutionGrid[op_cell.r][op_cell.c].value = op;
          solutionGrid[n2_cell.r][n2_cell.c].value             =
num2.toString();
          solutionGrid[n2_cell.r][n2_cell.c].type = 'number';
          if           (eq_cell.value           ===           'e')
solutionGrid[eq_cell.r][eq_cell.c].value = '=';
          solutionGrid[res_cell.r][res_cell.c].value           =
result.toString();
          solutionGrid[res_cell.r][res_cell.c].type = 'number';
        }

        if (success) {
          userGrid      =      Array(size).fill(0).map(()      =>
Array(size).fill(null));
          for (let r = 0; r < size; r++) {
            for (let c = 0; c < size; c++) {
              const cell = solutionGrid[r][c];
              if (cell.type === 'number') {
                const     isHidden     =     Math.random()     <
template.blanksRatio;
                userGrid[r][c] = { ...cell, value: isHidden ? ''
: cell.value, isHidden: isHidden };
              } else {
                userGrid[r][c] = { ...cell };
              }
            }
          }
          return { userGrid, solutionGrid, size };
        }
        attempts++;
      }
      console.error("Failed to generate a Mathecross puzzle after
max attempts.");
      return null;
    };

    const  MathecrossGrid  =  ({  grid,  size,  handleCellChange,
solutionGrid, isSolved }) => {
      return (
        <div
```

```
          className="grid gap-0.5 p-1 bg-gray-200 dark:bg-gray-
700 rounded-lg shadow-inner"
          style={{
            gridTemplateColumns:   `repeat(${size},   minmax(0,
1fr))`,
            gridTemplateRows: `repeat(${size}, minmax(0, 1fr))`,
            width: 'min(95vw, 500px)',
            height: 'min(95vw, 500px)',
          }}
        >
          {grid.map((row, rIdx) =>
            row.map((cell, cIdx) => {
              const isBlackCell = cell.type === 'black';
              const  isFixedCell  =  cell.type  ===  'operator'  ||
cell.type === 'equals';
              const  isNumberInput  =  cell.type  ===  'number'  &&
cell.isHidden;

              const  isCorrect  =  isSolved  ||  (isBlackCell  ||
isFixedCell || (cell.value === '' && cell.isHidden) || (cell.value
=== solutionGrid[rIdx][cIdx].value));
              const inputClass = isCorrect
                ? 'text-gray-900 dark:text-white'
                : 'text-red-500 font-bold';

              return (
                <div
                  key={`${rIdx}-${cIdx}`}
                  className={`relative     flex     items-center
justify-center rounded-lg overflow-hidden
                    ${isBlackCell  ?  'bg-gray-700  dark:bg-gray-
900' : (isFixedCell ? 'bg-blue-200 dark:bg-blue-900' : 'bg-white
dark:bg-gray-800')}
                    border  border-gray-400  dark:border-gray-600
font-semibold`}
                >
                  {isBlackCell ? (
                    ''
                  ) : isFixedCell ? (
                    <span  className="text-xl  sm:text-2xl  text-
gray-800 dark:text-gray-200">
                      {cell.value}
                    </span>
                  ) : (
                    <input
                      type="number"
                      min="1"
                      max="9"
                      value={cell.value}
                      onChange={(e)   =>   handleCellChange(rIdx,
cIdx, e.target.value)}
```

```
                        className={`w-full   h-full   text-center
text-2xl    sm:text-4xl    bg-transparent    focus:outline-none
focus:ring-2 focus:ring-blue-500 rounded-lg ${inputClass}`}
                        readOnly={!isNumberInput || isSolved}
                      />
                    )}
                  </div>
                );
              })
            )}
          </div>
        );
      };

     const MathecrossGame = ({ db, auth, userId, setView,
difficulty, setGameCompleteTime, setGameDifficulty }) => {
        const [size, setSize] = useState(0);
        const [grid, setGrid] = useState([]);
        const [solutionGrid, setSolutionGrid] = useState([]);
        const [timer, setTimer] = useState(0);
        const [isRunning, setIsRunning] = useState(false);
        const [hintsLeft, setHintsLeft] = useState(3);
        const [isSolved, setIsSolved] = useState(false);
        const [modalOpen, setModalOpen] = useState(false);
        const [modalContent, setModalContent] = useState('');

        const timerRef = useRef(null);

        const startTimer = () => {
          if (!isRunning) {
            setIsRunning(true);
            timerRef.current = setInterval(() => {
              setTimer((prevTime) => prevTime + 1);
            }, 1000);
          }
        };

        const stopTimer = () => {
          setIsRunning(false);
          clearInterval(timerRef.current);
        };

        const generateNewPuzzle = useCallback(() => {
          const puzzle = createMathecrossPuzzle(difficulty);
          if (puzzle) {
            setGrid(puzzle.userGrid);
            setSolutionGrid(puzzle.solutionGrid);
            setSize(puzzle.size);
            setTimer(0);
            setHintsLeft(3);
            setIsSolved(false);
            stopTimer();
            startTimer();
```

```
        } else {
          setModalContent('Не вдалося згенерувати головоломку.
Спробуйте інший рівень складності.');
          setModalOpen(true);
        }
      }, [difficulty]);

      useEffect(() => {
        generateNewPuzzle();
      }, [generateNewPuzzle]);

      useEffect(() => {
        if (isSolved) {
          stopTimer();
          setModalContent('Вітаємо!        Ви        розв\'язали
головоломку!');
          setModalOpen(true);
          setGameCompleteTime(timer);
          setGameDifficulty(difficulty);
        }
      },       [isSolved,       timer,       setGameCompleteTime,
setGameDifficulty, difficulty]);

      const handleCellChange = (r, c, value) => {
        if   (isSolved   ||   grid[r][c].type   !==   'number'   ||
!grid[r][c].isHidden) return;

        const newGrid = JSON.parse(JSON.stringify(grid));
        const    numValue    =    value    ===    ''    ?    ''    :
parseInt(value).toString();

        if ((numValue >= '1' && numValue <= '9') || numValue ===
'') {
          newGrid[r][c].value = numValue;
          setGrid(newGrid);
          checkIfSolved(newGrid);
        }
      };

      const checkIfSolved = (currentGrid) => {
        let allCellsFilledAndCorrect = true;
        const template = mathecrossPuzzleTemplates[difficulty];

        for (let r = 0; r < size; r++) {
          for (let c = 0; c < size; c++) {
            const cell = currentGrid[r][c];
            if (cell.type === 'number') {
              if   (cell.value   ===   ''   ||   cell.value   !==
solutionGrid[r][c].value) {
                allCellsFilledAndCorrect = false;
                break;
              }
            }
```

```
          }
          if (!allCellsFilledAndCorrect) break;
        }

        if (!allCellsFilledAndCorrect) return;

        let allExpressionsValid = true;
        for (const eqDef of template.equations) {
          const [n1_coord, op_coord, n2_coord, eq_coord,
res_coord] = eqDef.cells;
          const n1 =
parseInt(currentGrid[n1_coord[0]][n1_coord[1]].value);
          const op =
currentGrid[op_coord[0]][op_coord[1]].value;
          const n2 =
parseInt(currentGrid[n2_coord[0]][n2_coord[1]].value);
          const res =
parseInt(currentGrid[res_coord[0]][res_coord[1]].value);

          let calculatedResult;
          switch (op) {
            case '+': calculatedResult = n1 + n2; break;
            case '-': calculatedResult = n1 - n2; break;
            case '*': calculatedResult = n1 * n2; break;
            case '/': calculatedResult = n1 / n2; break;
            default: calculatedResult = NaN;
          }

          if (isNaN(calculatedResult) || calculatedResult !== res
|| !Number.isInteger(calculatedResult) || calculatedResult < 1 ||
calculatedResult > 9) {
            allExpressionsValid = false;
            break;
          }
        }

        if (allExpressionsValid) {
          setIsSolved(true);
        }
      };


      const applyHint = () => {
        if (hintsLeft > 0 && !isSolved) {
          let incorrectOrEmptyCells = [];
          for (let r = 0; r < size; r++) {
            for (let c = 0; c < size; c++) {
              const cell = grid[r][c];
              if (cell.type === 'number' && cell.isHidden &&
cell.value !== solutionGrid[r][c].value) {
                incorrectOrEmptyCells.push({ r, c });
              }
            }
```

```
        }

        if (incorrectOrEmptyCells.length > 0) {
          const hintCell =
incorrectOrEmptyCells[Math.floor(Math.random() *
incorrectOrEmptyCells.length)];
          const newGrid = JSON.parse(JSON.stringify(grid));
          newGrid[hintCell.r][hintCell.c].value =
solutionGrid[hintCell.r][hintCell.c].value;
          newGrid[hintCell.r][hintCell.c].isHidden = false;
          setGrid(newGrid);
          setHintsLeft(hintsLeft - 1);
          checkIfSolved(newGrid);
        } else {
          setModalContent('Немає порожніх або неправильних
клітинок для підказки.');
          setModalOpen(true);
        }
      } else {
        setModalContent('У вас більше немає підказок або
головоломка вже розв\'язана.');
        setModalOpen(true);
      }
    };

    return (
      <div className="flex flex-col items-center justify-center
p-4">
        <h2 className="text-3xl font-bold text-blue-600
dark:text-blue-400 mb-4">Математичний Кросворд</h2>
        <p className="text-xl text-gray-700 dark:text-gray-300
mb-4">Складність: {difficulty}</p>
        <Timer time={timer} />
        {grid.length > 0 && (
          <MathecrossGrid
            grid={grid}
            size={size}
            handleCellChange={handleCellChange}
            solutionGrid={solutionGrid}
            isSolved={isSolved}
          />
        )}
        <div className="mt-6 flex flex-wrap justify-center gap-
4">
          <button
            onClick={applyHint}
            className="px-6 py-3 bg-green-500 hover:bg-green-
600 text-white rounded-lg shadow-md transition-colors
disabled:opacity-50 disabled:cursor-not-allowed"
            disabled={hintsLeft === 0 || isSolved}
          >
            Підказки ({hintsLeft}/3)
          </button>
```

```
            <button
              onClick={generateNewPuzzle}
              className="px-6 py-3 bg-purple-500 hover:bg-
purple-600 text-white rounded-lg shadow-md transition-colors"
            >
              Нова головоломка
            </button>
            <button
              onClick={() => { setView('mainMenu'); stopTimer();
}}
              className="px-6 py-3 bg-gray-500 hover:bg-gray-600
text-white rounded-lg shadow-md transition-colors"
            >
              Головне меню
            </button>
          </div>
          <Modal isOpen={modalOpen} onClose={() =>
setModalOpen(false)}>
            <p className="text-lg">{modalContent}</p>
          </Modal>
        </div>
      );
    };

    const MathecrossProfile = ({ db, userId, setView }) => {
      const [profileData, setProfileData] = useState(null);
      const [loading, setLoading] = useState(true);

      useEffect(() => {
        if (!db || !userId) return;
        const userProfileRef = doc(db,
`artifacts/${appId}/users/${userId}/mathecross_profile`,
'stats');
        const unsubscribe = onSnapshot(userProfileRef, (docSnap)
=> {
          if (docSnap.exists()) {
setProfileData(docSnap.data()); } else { setProfileData({
bestTimes: {} }); }
          setLoading(false);
        }, (error) => { console.error("Помилка отримання даних
профілю:", error); setLoading(false); });
        return () => unsubscribe();
      }, [db, userId]);

      if (loading) { return <div className="text-center text-
gray-700 dark:text-gray-300 p-8">Завантаження профілю...</div>; }
      return (
        <div className="p-8 flex flex-col items-center min-h-
screen">
          <h2 className="text-4xl font-bold text-blue-600
dark:text-blue-400 mb-8">Профіль Математичного Кросворду</h2>
```

```
          <p className="text-xl text-gray-800 dark:text-gray-200
mb-6">Ваш ID: <span className="font-mono bg-gray-200 dark:bg-gray-
700 p-2 rounded-md break-all">{userId || 'Недоступно'}</span></p>
          {profileData?.bestTimes                                  &&
Object.keys(profileData.bestTimes).length > 0 ? (
            <div   className="bg-white   dark:bg-gray-800   p-6
rounded-lg shadow-lg w-full max-w-md">
              <h3 className="text-2xl font-semibold text-gray-
800 dark:text-gray-200 mb-4 text-center">Ваші найкращі часи</h3>
              <ul className="space-y-2">

{Object.entries(profileData.bestTimes).sort(([diffA], [diffB]) =>
{
                    const order = ['Новачок', 'Досвідчений',
'Професіонал', 'Експерт', 'Геній'];
                    return        order.indexOf(diffA)        -
order.indexOf(diffB);
                  }).map(([difficulty, time]) => (
                  <li key={difficulty} className="flex justify-
between items-center bg-gray-100 dark:bg-gray-700 p-3 rounded-md">
                    <span  className="font-medium  text-gray-700
dark:text-gray-300">{difficulty}:</span>
                    <span   className="text-blue-600   dark:text-
blue-400 font-bold">{Math.floor(time / 60).toString().padStart(2,
'0')}:{(time % 60).toString().padStart(2, '0')}</span>
                  </li>
                ))}
              </ul>
            </div>
          ) : (
            <p className="text-lg text-gray-600 dark:text-gray-
400">Поки що немає даних про найкращі часи. Почніть грати, щоб
зберегти свій результат!</p>
          )}
          <button   onClick={()   =>   setView('profileSelect')}
className="mt-8 px-8 py-3 bg-blue-500 hover:bg-blue-600 text-white
rounded-lg  shadow-lg  transition-colors">Повернутися  до  вибору
профілю</button>
        </div>
      );
    };

    const MathecrossLeaderboard = ({ db, setView }) => {
      const [leaderboardData, setLeaderboardData] = useState([]);
      const [loading, setLoading] = useState(true);

      useEffect(() => {
        if (!db) return;
        const    leaderboardCollectionRef    =    collection(db,
`artifacts/${appId}/public/data/mathecross_leaderboard`);
        const fetchLeaderboard = async () => {
          try {
```

```
            const querySnapshot = await
getDocs(leaderboardCollectionRef);
            let data = querySnapshot.docs.map(doc => ({ id:
doc.id, ...doc.data() }));
            data.sort((a, b) => a.time - b.time);
            setLeaderboardData(data.slice(0, 10));
          } catch (error) { console.error("Помилка отримання
рейтингу:", error); } finally { setLoading(false); }
        };
        const unsubscribe = onSnapshot(leaderboardCollectionRef,
(snapshot) => {
          let data = snapshot.docs.map(doc => ({ id: doc.id,
...doc.data() }));
          data.sort((a, b) => a.time - b.time);
          setLeaderboardData(data.slice(0, 10));
          setLoading(false);
        }, (error) => { console.error("Помилка отримання
рейтингу:", error); setLoading(false); });
        fetchLeaderboard();
        return () => unsubscribe();
      }, [db]);

      if (loading) { return <div className="text-center text-
gray-700 dark:text-gray-300 p-8">Завантаження рейтингу...</div>;
}
      return (
        <div className="p-8 flex flex-col items-center min-h-
screen">
          <h2 className="text-4xl font-bold text-blue-600
dark:text-blue-400 mb-8">Рейтинг Математичного Кросворду</h2>
          {leaderboardData.length > 0 ? (
            <div className="bg-white dark:bg-gray-800 p-6
rounded-lg shadow-lg w-full max-w-2xl">
              <table className="min-w-full divide-y divide-gray-
200 dark:divide-gray-700">
                <thead className="bg-gray-50 dark:bg-gray-700">
                  <tr>
                    <th className="px-6 py-3 text-left text-xs
font-medium text-gray-500 dark:text-gray-300 uppercase tracking-
wider">№</th>
                    <th className="px-6 py-3 text-left text-xs
font-medium text-gray-500 dark:text-gray-300 uppercase tracking-
wider">Складність</th>
                    <th className="px-6 py-3 text-left text-xs
font-medium text-gray-500 dark:text-gray-300 uppercase tracking-
wider">Час</th>
                    <th className="px-6 py-3 text-left text-xs
font-medium text-gray-500 dark:text-gray-300 uppercase tracking-
wider">Користувач ID</th>
                  </tr>
                </thead>
                <tbody className="bg-white dark:bg-gray-900
divide-y divide-gray-200 dark:divide-gray-700">
```

```
                  {leaderboardData.map((entry, index) => (
                    <tr key={entry.id}>
                      <td className="px-6 py-4 whitespace-nowrap
text-sm font-medium text-gray-900 dark:text-gray-100">{index +
1}</td>
                      <td className="px-6 py-4 whitespace-nowrap
text-sm text-gray-700 dark:text-gray-300">{entry.difficulty}</td>
                      <td className="px-6 py-4 whitespace-nowrap
text-sm       text-blue-600       dark:text-blue-400       font-
bold">{Math.floor(entry.time     /     60).toString().padStart(2,
'0')}:{(entry.time % 60).toString().padStart(2, '0')}</td>
                      <td className="px-6 py-4 whitespace-nowrap
text-sm       text-gray-500       dark:text-gray-400       break-
all">{entry.userId}</td>
                    </tr>
                  ))}
                </tbody>
              </table>
            </div>
          ) : (
            <p className="text-lg text-gray-600 dark:text-gray-
400">Рейтинг поки що порожній. Будьте першим!</p>
          )}
          <button onClick={() => setView('leaderboardSelect')}
className="mt-8 px-8 py-3 bg-blue-500 hover:bg-blue-600 text-white
rounded-lg shadow-lg transition-colors">Повернутися до вибору
рейтингу</button>
        </div>
      );
    };

    const MathecrossHelp = ({ setView }) => {
      return (
        <div  className="p-8  flex  flex-col  items-center  min-h-
screen">
          <h2   className="text-4xl   font-bold   text-blue-600
dark:text-blue-400    mb-8">Довідка:    Правила    Математичного
Кросворду</h2>
          <div className="bg-white dark:bg-gray-800 p-6 rounded-
lg shadow-lg max-w-3xl text-gray-800 dark:text-gray-200 leading-
relaxed space-y-4">
            <p>Математичний   кросворд   (Mathecross)   —   це
головоломка, де сітка клітин схожа на традиційні кросворди, але в
клітинах записані числа, знаки математичних операцій (+, -, *, /)
та знак рівності (=), які разом утворюють математичні вирази. Мета
гри полягає в тому, щоб заповнити порожні клітини числами від 1 до
9 так, щоб усі вирази по горизонталі та по вертикалі були
правильними.</p>
            <ul className="list-disc list-inside space-y-2">
              <li>Кожен вираз складається з п'яти клітин: число,
оператор,   число,   знак   рівності,   результат   (наприклад,
$2+3=5$).</li>
```

```
                <li>Вирази перетинаються між собою. Число в
перетині клітини повинно задовольняти обидва вирази, в яких воно
використовується.</li>
                <li>Всі числа, які ви вводите, повинні бути від 1
до 9.</li>
                <li>Результати операцій також повинні бути цілими
числами в діапазоні від 1 до 9.</li>
                <li>Пусті клітини потрібно заповнити так, щоб усі
рівняння були вірними.</li>
              </ul>
              <p className="font-bold">Приклад:</p>
              <pre  className="bg-gray-100  dark:bg-gray-700  p-3
rounded-md text-sm font-mono whitespace-pre-wrap">
                2 + 3 = 5
                    *
                    2
                    =
                    6
              </pre>
              <p>У цьому прикладі число $3$ є частиною як
горизонтального виразу ($2+3=5$), так і вертикального виразу
($3*2=6$).</p>
            </div>
            <button    onClick={()    =>    setView('helpSelect')}
className="mt-8 px-8 py-3 bg-blue-500 hover:bg-blue-600 text-white
rounded-lg  shadow-lg  transition-colors">Повернутися  до  вибору
довідки</button>
          </div>
        );
      };

      // --- KenKen Components (Copied from previous version) ---
      const generateLatinSquare = (size) => {
        const     grid     =     Array(size).fill(0).map(()     =>
Array(size).fill(0));
        const numbers = Array.from({ length: size }, (_, i) => i +
1);
        for (let r = 0; r < size; r++) {
          for (let c = 0; c < size; c++) {
            grid[r][c] = numbers[(r + c) % size];
          }
        }
        return grid;
      };

      const shuffleArray = (array) => {
        for (let i = array.length - 1; i > 0; i--) {
          const j = Math.floor(Math.random() * (i + 1));
          [array[i], array[j]] = [array[j], array[i]];
        }
      };

      const createKenKenCages = (solutionGrid, difficulty) => {
```

```
      const size = solutionGrid.length;
      const    visited    =    Array(size).fill(0).map(()    =>
Array(size).fill(false));
      const cages = [];
      const operations = ['+', '*', '-', '/'];

      const  maxCageSize  =  difficulty  ===  'Новачок'  ?  3  :
(difficulty === 'Досвідчений' ? 4 : (difficulty === 'Професіонал'
? 5 : (difficulty === 'Експерт' ? 6 : size)));

      for (let r = 0; r < size; r++) {
        for (let c = 0; c < size; c++) {
          if (!visited[r][c]) {
            const cageCells = [{ r, c }];
            visited[r][c] = true;
            const cellsToVisit = [{ r, c }];
            shuffleArray(cellsToVisit);
            let currentCageSize = 1;

            while (cellsToVisit.length > 0 && currentCageSize <
maxCageSize) {
              const   {   r:   currentR,   c:   currentC   }   =
cellsToVisit.pop();
              const neighbors = [
                { r: currentR - 1, c: currentC },
                { r: currentR + 1, c: currentC },
                { r: currentR, c: currentC - 1 },
                { r: currentR, c: currentC + 1 },
              ].filter(n =>
                n.r >= 0 && n.r < size && n.c >= 0 && n.c < size
&& !visited[n.r][n.c]
              );
              shuffleArray(neighbors);
              if (neighbors.length > 0) {
                const nextCell = neighbors[0];
                cageCells.push(nextCell);
                visited[nextCell.r][nextCell.c] = true;
                cellsToVisit.push(nextCell);
                currentCageSize++;
              }
            }
            let operation, target;
            let validOperationFound = false;

            while (!validOperationFound) {
              if (cageCells.length === 1) {
                operation = '';
                target                                          =
solutionGrid[cageCells[0].r][cageCells[0].c];
                validOperationFound = true;
              } else if (cageCells.length === 2) {
                const                    val1                   =
solutionGrid[cageCells[0].r][cageCells[0].c];
```

```
                const                    val2                    =
solutionGrid[cageCells[1].r][cageCells[1].c];
                const opsForTwoCells = operations.filter(op => op
!== '+' && op !== '*');
                operation                                        =
opsForTwoCells[Math.floor(Math.random()                          *
opsForTwoCells.length)];

                if (operation === '-') {
                  target = Math.abs(val1 - val2);
                  if (target !== 0) validOperationFound = true;
                } else if (operation === '/') {
                  const  [larger,  smaller]  =  [Math.max(val1,
val2), Math.min(val1, val2)];
                  if (smaller !== 0 && larger % smaller === 0) {
                    target = larger / smaller;
                    validOperationFound = true;
                  }
                } else {
                     operation                                   =
operations[Math.floor(Math.random() * operations.length)];
                     if (operation === '+') {
                        target = cageCells.reduce((sum, { r, c })
=> sum + solutionGrid[r][c], 0);
                        validOperationFound = true;
                     } else if (operation === '*') {
                        target = cageCells.reduce((prod, { r, c
}) => prod * solutionGrid[r][c], 1);
                        validOperationFound = true;
                     }
                }
                if (!validOperationFound) {
                     operation = Math.random() < 0.5 ? '+' : '*';
                     if (operation === '+') {
                        target = cageCells.reduce((sum, { r, c })
=> sum + solutionGrid[r][c], 0);
                        validOperationFound = true;
                     } else {
                        target = cageCells.reduce((prod, { r, c
}) => prod * solutionGrid[r][c], 1);
                        validOperationFound = true;
                     }
                }
              } else {
                operation  =  operations[Math.floor(Math.random()
* 2)];
                if (operation === '+') {
                  target = cageCells.reduce((sum, { r, c }) =>
sum + solutionGrid[r][c], 0);
                  validOperationFound = true;
                } else if (operation === '*') {
                  target = cageCells.reduce((prod, { r, c }) =>
prod * solutionGrid[r][c], 1);
```

```
                    if (target < 10000 && target > 0) {
                      validOperationFound = true;
                    } else {
                      operation = '+';
                      target = cageCells.reduce((sum, { r, c }) =>
sum + solutionGrid[r][c], 0);
                      validOperationFound = true;
                    }
                  }
                }
              }
              cages.push({ cells: cageCells, operation, target });
            }
          }
        }
        return cages;
      };

      const getKenKenBorders = (r, c, size, cages) => {
        const borders = { top: false, right: false, bottom: false,
left: false };
        const cage = cages.find(cage =>
          cage.cells.some(cell => cell.r === r && cell.c === c)
        );
        if (!cage) return borders;
        if (r === 0 || !cage.cells.some(cell => cell.r === r - 1 &&
cell.c === c)) { borders.top = true; }
        if (c === size - 1 || !cage.cells.some(cell => cell.r ===
r && cell.c === c + 1)) { borders.right = true; }
        if (r === size - 1 || !cage.cells.some(cell => cell.r ===
r + 1 && cell.c === c)) { borders.bottom = true; }
        if (c === 0 || !cage.cells.some(cell => cell.r === r &&
cell.c === c - 1)) { borders.left = true; }
        return borders;
      };

      const KenKenGrid = ({ grid, cages, size, handleCellChange,
solutionGrid, isSolved }) => {
        return (
          <div
            className="grid gap-0.5 p-1 bg-gray-200 dark:bg-gray-
700 rounded-lg shadow-inner"
            style={{
              gridTemplateColumns:   `repeat(${size},   minmax(0,
1fr))`,
              gridTemplateRows: `repeat(${size}, minmax(0, 1fr))`,
              width: 'min(95vw, 500px)',
              height: 'min(95vw, 500px)',
            }}
          >
            {grid.map((row, rIdx) =>
              row.map((cell, cIdx) => {
                const cage = cages.find(c =>
```

```
                c.cells.some(pos => pos.r === rIdx && pos.c ===
cIdx)
              );
              const isTopLeftOfCage = cage && cage.cells[0].r ===
rIdx && cage.cells[0].c === cIdx;
              const borders = getKenKenBorders(rIdx, cIdx, size,
cages);

              const   isCorrect   =   isSolved   ||   (cell   ===
solutionGrid[rIdx][cIdx] || cell === '');
              const inputClass = isCorrect
                ? 'text-gray-900 dark:text-white'
                : 'text-red-500 font-bold';

              return (
                <div
                  key={`${rIdx}-${cIdx}`}
                  className={`relative     flex     items-center
justify-center bg-white dark:bg-gray-900 overflow-hidden ${
                    borders.top  ?  'border-t-2  border-gray-900
dark:border-gray-100' : ''
                  } ${
                    borders.right ? 'border-r-2 border-gray-900
dark:border-gray-100' : ''
                  } ${
                    borders.bottom ? 'border-b-2 border-gray-900
dark:border-gray-100' : ''
                  } ${
                    borders.left  ?  'border-l-2  border-gray-900
dark:border-gray-100' : ''
                  }`}
                >
                  {isTopLeftOfCage && (
                    <div className="absolute top-0 left-0 text-
xs sm:text-sm font-semibold p-1 text-gray-700 dark:text-gray-300
pointer-events-none">
                      {cage.target} {cage.operation}
                    </div>
                  )}
                  <input
                    type="number"
                    min="1"
                    max={size}
                    value={cell === 0 ? '' : cell}
                    onChange={(e)    =>    handleCellChange(rIdx,
cIdx, e.target.value)}
                    className={`w-full h-full text-center text-
2xl  sm:text-4xl  bg-transparent  focus:outline-none  focus:ring-2
focus:ring-blue-500 rounded-lg ${inputClass}`}
                    readOnly={isSolved}
                  />
                </div>
              );
```

```
            })
          )}
        </div>
      );
    };

    const KenKenGame = ({ db, auth, userId, setView, difficulty,
setGameCompleteTime, setGameDifficulty }) => {
      const [size, setSize] = useState(0);
      const [grid, setGrid] = useState([]);
      const [solutionGrid, setSolutionGrid] = useState([]);
      const [cages, setCages] = useState([]);
      const [timer, setTimer] = useState(0);
      const [isRunning, setIsRunning] = useState(false);
      const [hintsLeft, setHintsLeft] = useState(3);
      const [isSolved, setIsSolved] = useState(false);
      const [modalOpen, setModalOpen] = useState(false);
      const [modalContent, setModalContent] = useState('');

      const timerRef = useRef(null);

      const startTimer = () => {
        if (!isRunning) {
          setIsRunning(true);
          timerRef.current = setInterval(() => {
            setTimer((prevTime) => prevTime + 1);
          }, 1000);
        }
      };
      const stopTimer = () => {
        setIsRunning(false);
        clearInterval(timerRef.current);
      };

      const getGridSize = useCallback((diff) => {
        switch (diff) {
          case 'Новачок': return 4;
          case 'Досвідчений': return 5;
          case 'Професіонал': return 6;
          case 'Експерт': return 7;
          case 'Геній': return 8;
          default: return 6;
        }
      }, []);

      const generateNewPuzzle = useCallback(() => {
        const newSize = getGridSize(difficulty);
        setSize(newSize);
        const newSolutionGrid = generateLatinSquare(newSize);
        const   newCages   =   createKenKenCages(newSolutionGrid,
difficulty);
```

```
        setGrid(Array(newSize).fill(0).map(() => Array(newSize).fill(0)));
        setSolutionGrid(newSolutionGrid);
        setCages(newCages);
        setTimer(0);
        setHintsLeft(3);
        setIsSolved(false);
        stopTimer();
        startTimer();
      }, [difficulty, getGridSize]);

      useEffect(() => {
        generateNewPuzzle();
      }, [generateNewPuzzle]);

      useEffect(() => {
        if (isSolved) {
          stopTimer();
          setModalContent('Вітаємо! Ви розв\'язали головоломку!');
          setModalOpen(true);
          setGameCompleteTime(timer);
          setGameDifficulty(difficulty);
        }
      }, [isSolved, timer, setGameCompleteTime, setGameDifficulty, difficulty]);

      const handleCellChange = (r, c, value) => {
        if (isSolved) return;
        const newGrid = [...grid];
        const numValue = value === '' ? 0 : parseInt(value);
        if (numValue >= 0 && numValue <= size) {
          newGrid[r][c] = numValue;
          setGrid(newGrid);
          checkIfSolved(newGrid);
        }
      };

      const checkIfSolved = (currentGrid) => {
        let allCellsFilled = true;
        for (let r = 0; r < size; r++) {
          for (let c = 0; c < size; c++) {
            if (currentGrid[r][c] === 0 || currentGrid[r][c] !== solutionGrid[r][c]) {
              allCellsFilled = false;
              break;
            }
          }
          if (!allCellsFilled) break;
        }
        if (allCellsFilled) { setIsSolved(true); }
      };
```

```
      const applyHint = () => {
        if (hintsLeft > 0 && !isSolved) {
          let emptyCells = [];
          for (let r = 0; r < size; r++) {
            for (let c = 0; c < size; c++) {
              if (grid[r][c] === 0 || grid[r][c] !==
solutionGrid[r][c]) {
                emptyCells.push({ r, c });
              }
            }
          }
          if (emptyCells.length > 0) {
            const hintCell = emptyCells[Math.floor(Math.random()
* emptyCells.length)];
            const newGrid = [...grid];
            newGrid[hintCell.r][hintCell.c] =
solutionGrid[hintCell.r][hintCell.c];
            setGrid(newGrid);
            setHintsLeft(hintsLeft - 1);
            checkIfSolved(newGrid);
          } else {
            setModalContent('Немає порожніх або неправильних
клітинок для підказки.');
            setModalOpen(true);
          }
        } else {
          setModalContent('У вас більше немає підказок або
головоломка вже розв\'язана.');
          setModalOpen(true);
        }
      };

      return (
        <div className="flex flex-col items-center justify-center
p-4">
          <h2 className="text-3xl font-bold text-blue-600
dark:text-blue-400 mb-4">КенКен</h2>
          <p className="text-xl text-gray-700 dark:text-gray-300
mb-4">Складність: {difficulty}</p>
          <Timer time={timer} />
          <KenKenGrid
            grid={grid}
            cages={cages}
            size={size}
            handleCellChange={handleCellChange}
            solutionGrid={solutionGrid}
            isSolved={isSolved}
          />
          <div className="mt-6 flex flex-wrap justify-center gap-
4">
            <button
              onClick={applyHint}
```

```
              className="px-6 py-3 bg-green-500 hover:bg-green-
600    text-white    rounded-lg    shadow-md    transition-colors
disabled:opacity-50 disabled:cursor-not-allowed"
              disabled={hintsLeft === 0 || isSolved}
            >
              Підказки ({hintsLeft}/3)
            </button>
            <button
              onClick={generateNewPuzzle}
              className="px-6   py-3   bg-purple-500   hover:bg-
purple-600 text-white rounded-lg shadow-md transition-colors"
            >
              Нова головоломка
            </button>
            <button
              onClick={() => { setView('mainMenu'); stopTimer();
}}
              className="px-6 py-3 bg-gray-500 hover:bg-gray-600
text-white rounded-lg shadow-md transition-colors"
            >
              Головне меню
            </button>
          </div>
          <Modal      isOpen={modalOpen}      onClose={()      =>
setModalOpen(false)}>
            <p className="text-lg">{modalContent}</p>
          </Modal>
        </div>
      );
    };

    const KenKenProfile = ({ db, userId, setView }) => {
      const [profileData, setProfileData] = useState(null);
      const [loading, setLoading] = useState(true);

      useEffect(() => {
        if (!db || !userId) return;
        const          userProfileRef          =          doc(db,
`artifacts/${appId}/users/${userId}/kenken_profile`, 'stats');
        const unsubscribe = onSnapshot(userProfileRef, (docSnap)
=> {
          if                 (docSnap.exists())                 {
setProfileData(docSnap.data());   }   else   {   setProfileData({
bestTimes: {} }); }
          setLoading(false);
        }, (error) => { console.error("Помилка отримання даних
профілю:", error); setLoading(false); });
        return () => unsubscribe();
      }, [db, userId]);

      if (loading) { return <div className="text-center text-
gray-700 dark:text-gray-300 p-8">Завантаження профілю...</div>; }
      return (
```

```
        <div className="p-8 flex flex-col items-center min-h-
screen">
          <h2 className="text-4xl font-bold text-blue-600
dark:text-blue-400 mb-8">Профіль КенКен</h2>
          <p className="text-xl text-gray-800 dark:text-gray-200
mb-6">Ваш ID: <span className="font-mono bg-gray-200 dark:bg-gray-
700 p-2 rounded-md break-all">{userId || 'Недоступно'}</span></p>
          {profileData?.bestTimes &&
Object.keys(profileData.bestTimes).length > 0 ? (
            <div className="bg-white dark:bg-gray-800 p-6
rounded-lg shadow-lg w-full max-w-md">
              <h3 className="text-2xl font-semibold text-gray-
800 dark:text-gray-200 mb-4 text-center">Ваші найкращі часи</h3>
              <ul className="space-y-2">

{Object.entries(profileData.bestTimes).sort(([diffA], [diffB]) =>
{
                  const order = ['Новачок', 'Досвідчений',
'Професіонал', 'Експерт', 'Геній'];
                  return order.indexOf(diffA) -
order.indexOf(diffB);
                }).map(([difficulty, time]) => (
                  <li key={difficulty} className="flex justify-
between items-center bg-gray-100 dark:bg-gray-700 p-3 rounded-md">
                    <span className="font-medium text-gray-700
dark:text-gray-300">{difficulty}:</span>
                    <span className="text-blue-600 dark:text-
blue-400 font-bold">{Math.floor(time / 60).toString().padStart(2,
'0')}:{(time % 60).toString().padStart(2, '0')}</span>
                  </li>
                ))}
              </ul>
            </div>
          ) : (
            <p className="text-lg text-gray-600 dark:text-gray-
400">Поки що немає даних про найкращі часи. Почніть грати, щоб
зберегти свій результат!</p>
          )}
          <button onClick={() => setView('profileSelect')}
className="mt-8 px-8 py-3 bg-blue-500 hover:bg-blue-600 text-white
rounded-lg shadow-lg transition-colors">Повернутися до вибору
профілю</button>
        </div>
      );
    };

    const KenKenLeaderboard = ({ db, setView }) => {
      const [leaderboardData, setLeaderboardData] = useState([]);
      const [loading, setLoading] = useState(true);

      useEffect(() => {
        if (!db) return;
```

```
        const leaderboardCollectionRef = collection(db, `artifacts/${appId}/public/data/kenken_leaderboard`);
        const fetchLeaderboard = async () => {
          try {
            const querySnapshot = await getDocs(leaderboardCollectionRef);
            let data = querySnapshot.docs.map(doc => ({ id: doc.id, ...doc.data() }));
            data.sort((a, b) => a.time - b.time);
            setLeaderboardData(data.slice(0, 10));
          } catch (error) { console.error("Помилка отримання рейтингу:", error); } finally { setLoading(false); }
        };
        const unsubscribe = onSnapshot(leaderboardCollectionRef, (snapshot) => {
          let data = snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() }));
          data.sort((a, b) => a.time - b.time);
          setLeaderboardData(data.slice(0, 10));
          setLoading(false);
        }, (error) => { console.error("Помилка отримання рейтингу:", error); setLoading(false); });
        fetchLeaderboard();
        return () => unsubscribe();
      }, [db]);

      if (loading) { return <div className="text-center text-gray-700 dark:text-gray-300 p-8">Завантаження рейтингу...</div>; }
      return (
        <div className="p-8 flex flex-col items-center min-h-screen">
          <h2 className="text-4xl font-bold text-blue-600 dark:text-blue-400 mb-8">Рейтинг КенКен</h2>
          {leaderboardData.length > 0 ? (
            <div className="bg-white dark:bg-gray-800 p-6 rounded-lg shadow-lg w-full max-w-2xl">
              <table className="min-w-full divide-y divide-gray-200 dark:divide-gray-700">
                <thead className="bg-gray-50 dark:bg-gray-700">
                  <tr>
                    <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 dark:text-gray-300 uppercase tracking-wider">№</th>
                    <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 dark:text-gray-300 uppercase tracking-wider">Складність</th>
                    <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 dark:text-gray-300 uppercase tracking-wider">Час</th>
                    <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 dark:text-gray-300 uppercase tracking-wider">Користувач ID</th>
```

```
                  </tr>
                </thead>
                <tbody   className="bg-white   dark:bg-gray-900
divide-y divide-gray-200 dark:divide-gray-700">
                  {leaderboardData.map((entry, index) => (
                    <tr key={entry.id}>
                      <td className="px-6 py-4 whitespace-nowrap
text-sm  font-medium  text-gray-900  dark:text-gray-100">{index  +
1}</td>
                      <td className="px-6 py-4 whitespace-nowrap
text-sm text-gray-700 dark:text-gray-300">{entry.difficulty}</td>
                      <td className="px-6 py-4 whitespace-nowrap
text-sm       text-blue-600       dark:text-blue-400       font-
bold">{Math.floor(entry.time     /     60).toString().padStart(2,
'0')}:{(entry.time % 60).toString().padStart(2, '0')}</td>
                      <td className="px-6 py-4 whitespace-nowrap
text-sm       text-gray-500       dark:text-gray-400       break-
all">{entry.userId}</td>
                    </tr>
                  ))}
                </tbody>
              </table>
            </div>
          ) : (
            <p className="text-lg text-gray-600 dark:text-gray-
400">Рейтинг поки що порожній. Будьте першим!</p>
          )}
          <button  onClick={()  =>  setView('leaderboardSelect')}
className="mt-8 px-8 py-3 bg-blue-500 hover:bg-blue-600 text-white
rounded-lg  shadow-lg  transition-colors">Повернутися  до  вибору
рейтингу</button>
        </div>
      );
    };

    const KenKenHelp = ({ setView }) => {
      return (
        <div  className="p-8  flex  flex-col  items-center  min-h-
screen">
          <h2   className="text-4xl   font-bold   text-blue-600
dark:text-blue-400 mb-8">Довідка: Правила KenKen</h2>
          <div className="bg-white dark:bg-gray-800 p-6 rounded-
lg shadow-lg max-w-3xl text-gray-800 dark:text-gray-200 leading-
relaxed space-y-4">
            <p>KenKen (також відомий як KenDoku, MathDoku або
Calcudoku) — це математична головоломка, схожа на Судоку. Мета
полягає в тому, щоб заповнити сітку числами так, щоб:</p>
            <ul className="list-disc list-inside space-y-2">
              <li>Кожен рядок містив кожне число від 1 до розміру
сітки (наприклад, від 1 до 6 для сітки 6x6) рівно один раз.</li>
              <li>Кожен стовпець містив кожне число від 1 до
розміру сітки рівно один раз.</li>
```

```
              <li>Кожна група клітинок, обведена жирною лінією (називається "клітка" або "камерою"), містила числа, які, якщо їх поєднати за допомогою вказаної математичної операції (+, -, ×, ÷), дають цільове число.</li>
              <li>У клітинках з однією клітинкою просто вказується число, яке потрібно помістити в цю клітинку.</li>
            </ul>
            <p className="font-semibold">Приклад:</p>
            <p>Клітка "6+" означає, що числа в цій клітці повинні додаватися до 6.</p>
            <p>Клітка "2÷" означає, що одне число в цій клітці, поділене на інше, дає 2.</p>
            <p>Клітка "3-" означає, що різниця між числами в цій клітці становить 3.</p>
            <p className="font-bold">Ключові моменти:</p>
            <ul className="list-disc list-inside space-y-2">
              <li>На відміну від Судоку, число може повторюватися в одній клітці, якщо ця клітка має кілька клітинок і не є одним рядком або стовпцем.</li>
              <li>Для операцій віднімання та ділення порядок чисел не має значення; використовується абсолютне значення різниці або результат ділення більшого числа на менше.</li>
            </ul>
          </div>
          <button onClick={() => setView('helpSelect')} className="mt-8 px-8 py-3 bg-blue-500 hover:bg-blue-600 text-white rounded-lg shadow-lg transition-colors">Повернутися до вибору довідки</button>
        </div>
      );
    };

    // --- Kakuro Components (Copied from previous version) ---
    const generateUniqueNumbersForRun = (length) => {
      if (length > 9 || length < 1) return { numbers: null, sum: 0 };
      const availableNumbers = Array.from({ length: 9 }, (_, i) => i + 1);
      let numbers = [];
      let sum = 0;
      for (let i = 0; i < length; i++) {
        if (availableNumbers.length === 0) return { numbers: null, sum: 0 };
        const randomIndex = Math.floor(Math.random() * availableNumbers.length);
        const num = availableNumbers.splice(randomIndex, 1)[0];
        numbers.push(num);
        sum += num;
      }
      return { numbers, sum };
    };

    const createKakuroPuzzle = (size, difficulty) => {
```

```
      const grid = Array(size).fill(0).map(() => Array(size).fill('w'));
      const solutionGrid = Array(size).fill(0).map(() => Array(size).fill(0));
      const clues = [];

      grid[0][0] = 'b';
      for (let i = 1; i < size; i++) {
        grid[0][i] = Math.random() < 0.6 ? 'b' : 'w';
        grid[i][0] = Math.random() < 0.6 ? 'b' : 'w';
      }

      const internalBlackCellDensity = difficulty === 'Новачок' ? 0.2 :
                                       (difficulty === 'Досвідчений' ? 0.25 :
                                       (difficulty === 'Професіонал' ? 0.3 :
                                       (difficulty === 'Експерт' ? 0.35 : 0.4)));

      for (let r = 1; r < size; r++) {
        for (let c = 1; c < size; c++) {
          if (Math.random() < internalBlackCellDensity) {
            grid[r][c] = 'b';
          }
        }
      }

      for (let r = 1; r < size - 1; r++) {
        for (let c = 1; c < size - 1; c++) {
          if (grid[r][c] === 'w' &&
              grid[r-1][c] === 'b' && grid[r+1][c] === 'b' &&
              grid[r][c-1] === 'b' && grid[r][c+1] === 'b') {
            grid[r][c] = 'b';
          }
        }
      }

      for (let r = 0; r < size; r++) {
        for (let c = 0; c < size; c++) {
          if (grid[r][c] === 'b' && c + 1 < size && grid[r][c + 1] === 'w') {
            let runCells = [];
            for (let k = c + 1; k < size && grid[r][k] === 'w'; k++) {
              runCells.push({ r, c: k });
            }
            if (runCells.length > 0) {
              const { numbers, sum } = generateUniqueNumbersForRun(runCells.length);
              if (numbers) {
```

```
                clues.push({ r: r, c: c, type: 'right', sum: sum
});
                numbers.forEach((num, index) => {

solutionGrid[runCells[index].r][runCells[index].c] = num;
                });
              }
            }
          }
        }

        for (let c = 0; c < size; c++) {
          for (let r = 0; r < size; r++) {
            if (grid[r][c] === 'b' && r + 1 < size && grid[r + 1][c]
=== 'w') {
              let runCells = [];
              for (let k = r + 1; k < size && grid[k][c] === 'w';
k++) {
                runCells.push({ r: k, c: c });
              }
              if (runCells.length > 0) {
                const      {      numbers,      sum      }      =
generateUniqueNumbersForRun(runCells.length);
                if (numbers) {
                  clues.push({ r: r, c: c, type: 'down', sum: sum
});
                  numbers.forEach((num, index) => {

solutionGrid[runCells[index].r][runCells[index].c] = num;
                  });
                }
              }
            }
          }
        }

        const    userGrid    =    Array(size).fill(0).map(()    =>
Array(size).fill(0));
        for (let r = 0; r < size; r++) {
          for (let c = 0; c < size; c++) {
            if (grid[r][c] === 'b') {
              userGrid[r][c] = 'b';
            } else {
              userGrid[r][c] = 0;
            }
          }
        }
        return { initialGrid: userGrid, solutionGrid, clues };
      };

      const KakuroGrid = ({ grid, clues, size, handleCellChange,
solutionGrid, isSolved }) => {
```

```
       return (
         <div
           className="grid gap-0.5 p-1 bg-gray-200 dark:bg-gray-
700 rounded-lg shadow-inner"
           style={{
             gridTemplateColumns:   `repeat(${size},   minmax(0,
1fr))`,
             gridTemplateRows: `repeat(${size}, minmax(0, 1fr))`,
             width: 'min(95vw, 500px)',
             height: 'min(95vw, 500px)',
           }}
         >
           {grid.map((row, rIdx) =>
             row.map((cell, cIdx) => {
               const isBlackCell = cell === 'b';
               const horizontalClue = clues.find(c => c.r === rIdx
&& c.c === cIdx && c.type === 'right');
               const verticalClue = clues.find(c => c.r === rIdx
&& c.c === cIdx && c.type === 'down');

               const isCorrect = isSolved || (isBlackCell || cell
=== 0 || cell === solutionGrid[rIdx][cIdx]);
               const inputClass = isCorrect
                 ? 'text-gray-900 dark:text-white'
                 : 'text-red-500 font-bold';

               return (
                 <div
                   key={`${rIdx}-${cIdx}`}
                   className={`relative     flex     items-center
justify-center rounded-lg overflow-hidden
                     ${isBlackCell  ?  'bg-gray-700  dark:bg-gray-
900' : 'bg-white dark:bg-gray-800'}
                     border   border-gray-400   dark:border-gray-
600`}
                 >
                   {isBlackCell ? (
                     <div className="relative w-full h-full flex
flex-col  items-center  justify-center  text-white  dark:text-gray-
200 text-xs sm:text-sm font-semibold p-1">
                       <div  className="absolute  top-0  left-0  w-
full h-full flex items-center justify-center">
                         {(verticalClue || horizontalClue) && (
                           <>
                             <div className="w-px h-full bg-gray-
400 dark:bg-gray-500 transform rotate-45 absolute"></div>
                             <div className="w-full h-px bg-gray-
400 dark:bg-gray-500 transform rotate-45 absolute"></div>
                           </>
                         )}
                       </div>
                       {verticalClue && (
```

```
                        <div className="absolute top-1 left-1/2
-translate-x-1/2 text-white dark:text-gray-200 text-xs sm:text-sm
font-semibold transform -rotate-45" style={{ transformOrigin: 'top
center' }}>
                          {verticalClue.sum}
                        </div>
                      )}
                      {horizontalClue && (
                        <div className="absolute bottom-1 right-
1/2 translate-x-1/2 text-white dark:text-gray-200 text-xs sm:text-
sm font-semibold transform -rotate-45" style={{ transformOrigin:
'bottom center' }}>
                          {horizontalClue.sum}
                        </div>
                      )}
                    </div>
                  ) : (
                    <input
                      type="number"
                      min="1"
                      max="9"
                      value={cell === 0 ? '' : cell}
                      onChange={(e)   =>   handleCellChange(rIdx,
cIdx, e.target.value)}
                      className={`w-full   h-full   text-center
text-2xl     sm:text-4xl     bg-transparent     focus:outline-none
focus:ring-2 focus:ring-blue-500 rounded-lg ${inputClass}`}
                      readOnly={isSolved}
                    />
                  )}
                </div>
              );
            })
          )}
        </div>
      );
    };

    const KakuroGame = ({ db, auth, userId, setView, difficulty,
setGameCompleteTime, setGameDifficulty }) => {
      const [size, setSize] = useState(0);
      const [grid, setGrid] = useState([]);
      const [solutionGrid, setSolutionGrid] = useState([]);
      const [clues, setClues] = useState([]);
      const [timer, setTimer] = useState(0);
      const [isRunning, setIsRunning] = useState(false);
      const [hintsLeft, setHintsLeft] = useState(3);
      const [isSolved, setIsSolved] = useState(false);
      const [modalOpen, setModalOpen] = useState(false);
      const [modalContent, setModalContent] = useState('');

      const timerRef = useRef(null);
```

```
        const startTimer = () => {
          if (!isRunning) {
            setIsRunning(true);
            timerRef.current = setInterval(() => {
              setTimer((prevTime) => prevTime + 1);
            }, 1000);
          }
        };
        const stopTimer = () => {
          setIsRunning(false);
          clearInterval(timerRef.current);
        };

        const getGridSize = useCallback((diff) => {
          switch (diff) {
            case 'Новачок': return 6;
            case 'Досвідчений': return 7;
            case 'Професіонал': return 8;
            case 'Експерт': return 9;
            case 'Геній': return 10;
            default: return 8;
          }
        }, []);

        const generateNewPuzzle = useCallback(() => {
          const newSize = getGridSize(difficulty);
          setSize(newSize);
          const { initialGrid, solutionGrid: newSolutionGrid,
clues: newClues } = createKakuroPuzzle(newSize, difficulty);

          setGrid(initialGrid);
          setSolutionGrid(newSolutionGrid);
          setClues(newClues);
          setTimer(0);
          setHintsLeft(3);
          setIsSolved(false);
          stopTimer();
          startTimer();
        }, [difficulty, getGridSize]);

        useEffect(() => {
          generateNewPuzzle();
        }, [generateNewPuzzle]);

        useEffect(() => {
          if (isSolved) {
            stopTimer();
            setModalContent('Вітаємо!        Ви        розв\'язали
головоломку!');
            setModalOpen(true);
            setGameCompleteTime(timer);
            setGameDifficulty(difficulty);
          }
```

```
    }, [isSolved, timer, setGameCompleteTime,
setGameDifficulty, difficulty]);

    const handleCellChange = (r, c, value) => {
      if (isSolved || grid[r][c] === 'b') return;
      const newGrid = [...grid.map(row => [...row])];
      const numValue = value === '' ? 0 : parseInt(value);
      if ((numValue >= 1 && numValue <= 9) || numValue === 0)
{
        newGrid[r][c] = numValue;
        setGrid(newGrid);
        checkIfSolved(newGrid);
      }
    };

    const checkIfSolved = (currentGrid) => {
      let allCellsFilledAndCorrect = true;
      for (let r = 0; r < size; r++) {
        for (let c = 0; c < size; c++) {
          if (currentGrid[r][c] === 'b') continue;
          if (currentGrid[r][c] === 0 || currentGrid[r][c] !==
solutionGrid[r][c]) {
            allCellsFilledAndCorrect = false;
            break;
          }
        }
        if (!allCellsFilledAndCorrect) break;
      }
      if (allCellsFilledAndCorrect) { setIsSolved(true); }
    };

    const applyHint = () => {
      if (hintsLeft > 0 && !isSolved) {
        let incorrectOrEmptyCells = [];
        for (let r = 0; r < size; r++) {
          for (let c = 0; c < size; c++) {
            if (grid[r][c] !== 'b' && (grid[r][c] === 0 ||
grid[r][c] !== solutionGrid[r][c])) {
              incorrectOrEmptyCells.push({ r, c });
            }
          }
        }
        if (incorrectOrEmptyCells.length > 0) {
          const hintCell =
incorrectOrEmptyCells[Math.floor(Math.random() *
incorrectOrEmptyCells.length)];
          const newGrid = [...grid.map(row => [...row])];
          newGrid[hintCell.r][hintCell.c] =
solutionGrid[hintCell.r][hintCell.c];
          setGrid(newGrid);
          setHintsLeft(hintsLeft - 1);
          checkIfSolved(newGrid);
        } else {
```

```
            setModalContent('Немає порожніх або неправильних
клітинок для підказки.');
            setModalOpen(true);
          }
        } else {
          setModalContent('У вас більше немає підказок або
головоломка вже розв\'язана.');
          setModalOpen(true);
        }
      };

      return (
        <div className="flex flex-col items-center justify-center
p-4">
          <h2 className="text-3xl font-bold text-blue-600
dark:text-blue-400 mb-4">Какуро</h2>
          <p className="text-xl text-gray-700 dark:text-gray-300
mb-4">Складність: {difficulty}</p>
          <Timer time={timer} />
          <KakuroGrid
            grid={grid}
            clues={clues}
            size={size}
            handleCellChange={handleCellChange}
            solutionGrid={solutionGrid}
            isSolved={isSolved}
          />
          <div className="mt-6 flex flex-wrap justify-center gap-
4">
            <button
              onClick={applyHint}
              className="px-6 py-3 bg-green-500 hover:bg-green-
600 text-white rounded-lg shadow-md transition-colors
disabled:opacity-50 disabled:cursor-not-allowed"
              disabled={hintsLeft === 0 || isSolved}
            >
              Підказки ({hintsLeft}/3)
            </button>
            <button
              onClick={generateNewPuzzle}
              className="px-6 py-3 bg-purple-500 hover:bg-
purple-600 text-white rounded-lg shadow-md transition-colors"
            >
              Нова головоломка
            </button>
            <button
              onClick={() => { setView('mainMenu'); stopTimer();
}}
              className="px-6 py-3 bg-gray-500 hover:bg-gray-600
text-white rounded-lg shadow-md transition-colors"
            >
              Головне меню
            </button>
```

```
          </div>
          <Modal      isOpen={modalOpen}      onClose={()      =>
setModalOpen(false)}>
            <p className="text-lg">{modalContent}</p>
          </Modal>
        </div>
      );
    };

    const KakuroProfile = ({ db, userId, setView }) => {
      const [profileData, setProfileData] = useState(null);
      const [loading, setLoading] = useState(true);

      useEffect(() => {
        if (!db || !userId) return;
        const          userProfileRef          =          doc(db,
`artifacts/${appId}/users/${userId}/kakuro_profile`, 'stats');
        const unsubscribe = onSnapshot(userProfileRef, (docSnap)
=> {
          if                 (docSnap.exists())                 {
setProfileData(docSnap.data());   }   else   {   setProfileData({
bestTimes: {} }); }
          setLoading(false);
        }, (error) => { console.error("Помилка отримання даних
профілю:", error); setLoading(false); });
        return () => unsubscribe();
      }, [db, userId]);

      if (loading) { return <div className="text-center text-
gray-700 dark:text-gray-300 p-8">Завантаження профілю...</div>; }
      return (
        <div  className="p-8  flex  flex-col  items-center  min-h-
screen">
          <h2   className="text-4xl   font-bold   text-blue-600
dark:text-blue-400 mb-8">Профіль Кауро</h2>
          <p className="text-xl text-gray-800 dark:text-gray-200
mb-6">Ваш ID: <span className="font-mono bg-gray-200 dark:bg-gray-
700 p-2 rounded-md break-all">{userId || 'Недоступно'}</span></p>
          {profileData?.bestTimes                                &&
Object.keys(profileData.bestTimes).length > 0 ? (
            <div   className="bg-white   dark:bg-gray-800   p-6
rounded-lg shadow-lg w-full max-w-md">
              <h3  className="text-2xl  font-semibold  text-gray-
800 dark:text-gray-200 mb-4 text-center">Ваші найкращі часи</h3>
              <ul className="space-y-2">

{Object.entries(profileData.bestTimes).sort(([diffA], [diffB]) =>
{
                  const  order  =  ['Новачок',  'Досвідчений',
'Професіонал', 'Експерт', 'Геній'];
                  return        order.indexOf(diffA)        -
order.indexOf(diffB);
                }).map(([difficulty, time]) => (
```

```
                  <li key={difficulty} className="flex justify-
between items-center bg-gray-100 dark:bg-gray-700 p-3 rounded-md">
                    <span className="font-medium text-gray-700
dark:text-gray-300">{difficulty}:</span>
                    <span  className="text-blue-600  dark:text-
blue-400 font-bold">{Math.floor(time / 60).toString().padStart(2,
'0')}:{(time % 60).toString().padStart(2, '0')}</span>
                  </li>
                ))}
              </ul>
            </div>
          ) : (
            <p className="text-lg text-gray-600 dark:text-gray-
400">Поки що немає даних про найкращі часи. Почніть грати, щоб
зберегти свій результат!</p>
          )}
          <button  onClick={()  =>  setView('profileSelect')}
className="mt-8 px-8 py-3 bg-blue-500 hover:bg-blue-600 text-white
rounded-lg shadow-lg transition-colors">Повернутися до вибору
профілю</button>
        </div>
      );
    };

    const KakuroLeaderboard = ({ db, setView }) => {
      const [leaderboardData, setLeaderboardData] = useState([]);
      const [loading, setLoading] = useState(true);

      useEffect(() => {
        if (!db) return;
        const   leaderboardCollectionRef   =   collection(db,
`artifacts/${appId}/public/data/kakuro_leaderboard`);
        const fetchLeaderboard = async () => {
          try {
            const         querySnapshot         =         await
getDocs(leaderboardCollectionRef);
            let  data  =  querySnapshot.docs.map(doc  =>  ({  id:
doc.id, ...doc.data() }));
            data.sort((a, b) => a.time - b.time);
            setLeaderboardData(data.slice(0, 10));
          } catch (error) { console.error("Помилка отримання
рейтингу:", error); } finally { setLoading(false); }
        };
        const unsubscribe = onSnapshot(leaderboardCollectionRef,
(snapshot) => {
          let  data  =  snapshot.docs.map(doc  =>  ({  id:  doc.id,
...doc.data() }));
          data.sort((a, b) => a.time - b.time);
          setLeaderboardData(data.slice(0, 10));
          setLoading(false);
        },   (error)   =>   {   console.error("Помилка   отримання
рейтингу:", error); setLoading(false); });
        fetchLeaderboard();
```

```
        return () => unsubscribe();
      }, [db]);

      if (loading) { return <div className="text-center text-
gray-700 dark:text-gray-300 p-8">Завантаження рейтингу...</div>;
}
      return (
        <div  className="p-8  flex  flex-col  items-center  min-h-
screen">
          <h2   className="text-4xl   font-bold   text-blue-600
dark:text-blue-400 mb-8">Рейтинг Кауро</h2>
          {leaderboardData.length > 0 ? (
            <div   className="bg-white   dark:bg-gray-800   p-6
rounded-lg shadow-lg w-full max-w-2xl">
              <table className="min-w-full divide-y divide-gray-
200 dark:divide-gray-700">
                <thead className="bg-gray-50 dark:bg-gray-700">
                  <tr>
                    <th  className="px-6  py-3  text-left  text-xs
font-medium text-gray-500 dark:text-gray-300 uppercase tracking-
wider">№</th>
                    <th  className="px-6  py-3  text-left  text-xs
font-medium text-gray-500 dark:text-gray-300 uppercase tracking-
wider">Складність</th>
                    <th  className="px-6  py-3  text-left  text-xs
font-medium text-gray-500 dark:text-gray-300 uppercase tracking-
wider">Час</th>
                    <th  className="px-6  py-3  text-left  text-xs
font-medium text-gray-500 dark:text-gray-300 uppercase tracking-
wider">Користувач ID</th>
                  </tr>
                </thead>
                <tbody   className="bg-white   dark:bg-gray-900
divide-y divide-gray-200 dark:divide-gray-700">
                  {leaderboardData.map((entry, index) => (
                    <tr key={entry.id}>
                      <td  className="px-6  py-4  whitespace-nowrap
text-sm  font-medium  text-gray-900  dark:text-gray-100">{index  +
1}</td>
                      <td  className="px-6  py-4  whitespace-nowrap
text-sm text-gray-700 dark:text-gray-300">{entry.difficulty}</td>
                      <td  className="px-6  py-4  whitespace-nowrap
text-sm       text-blue-600       dark:text-blue-400       font-
bold">{Math.floor(entry.time     /     60).toString().padStart(2,
'0')}:{(entry.time % 60).toString().padStart(2, '0')}</td>
                      <td  className="px-6  py-4  whitespace-nowrap
text-sm       text-gray-500       dark:text-gray-400       break-
all">{entry.userId}</td>
                    </tr>
                  ))}
                </tbody>
              </table>
            </div>
```

```
          ) : (
            <p className="text-lg text-gray-600 dark:text-gray-
400">Рейтинг поки що порожній. Будьте першим!</p>
          )}
          <button onClick={() => setView('leaderboardSelect')}
className="mt-8 px-8 py-3 bg-blue-500 hover:bg-blue-600 text-white
rounded-lg shadow-lg transition-colors">Повернутися до вибору
рейтингу</button>
        </div>
      );
    };

    const KakuroHelp = ({ setView }) => {
      return (
        <div className="p-8 flex flex-col items-center min-h-
screen">
          <h2 className="text-4xl font-bold text-blue-600
dark:text-blue-400 mb-8">Довідка: Правила Кауро</h2>
          <div className="bg-white dark:bg-gray-800 p-6 rounded-
lg shadow-lg max-w-3xl text-gray-800 dark:text-gray-200 leading-
relaxed space-y-4">
            <p>Кауро (також відомий як Cross Sums або Kakuro) —
це логічна головоломка, яка поєднує елементи Судоку, кросвордів та
базових математичних операцій. Мета полягає в тому, щоб заповнити
всі порожні клітинки числами від 1 до 9, дотримуючись наступних
правил:</p>
            <ul className="list-disc list-inside space-y-2">
              <li>Кожен рядок і кожен стовпець містять "блоки"
білих клітинок, розділені чорними клітинками. Ці блоки називаються
"пробігами".</li>
              <li>Кожен пробіг має числову "підказку", яка вказує
на суму всіх чисел у цьому пробігу. Горизонтальні підказки
знаходяться в чорних клітинках ліворуч від пробігу, а вертикальні
підказки — у чорних клітинках над пробігом.</li>
              <li>У межах одного пробігу (як горизонтального, так
і вертикального) усі числа повинні бути унікальними. Тобто, жодне
число не може повторюватися в одному пробігу.</li>
              <li>Заповнюються лише білі клітинки. Чорні клітинки
слугують лише для відображення підказок або як роздільники.</li>
            </ul>
            <p className="font-semibold">Приклад:</p>
            <p>Якщо чорна клітинка містить "10 →", це означає, що
сума чисел у білому пробігу праворуч від неї повинна дорівнювати
10.</p>
            <p>Якщо чорна клітинка містить "12 ↓", це означає, що
сума чисел у білому пробігу під нею повинна дорівнювати 12.</p>
            <p className="font-bold">Ключові моменти:</p>
            <ul className="list-disc list-inside space-y-2">
              <li>Числа від 1 до 9.</li>
              <li>Унікальні числа в кожному пробігу.</li>
              <li>Сума чисел у пробігу повинна дорівнювати
підказці.</li>
            </ul>
```

```
          </div>
          <button    onClick={()    =>    setView('helpSelect')}
className="mt-8 px-8 py-3 bg-blue-500 hover:bg-blue-600 text-white
rounded-lg  shadow-lg  transition-colors">Повернутися  до  вибору
довідки</button>
        </div>
      );
    };


    // --- Main App Component (Hub) ---
    const App = () => {
      const [theme, setTheme] = useState('light');
      const [currentView, setCurrentView] = useState('mainMenu');
//        'mainMenu',        'gameSelect',        'profileSelect',
'leaderboardSelect', 'helpSelect', game views
      const    [currentDifficulty,    setCurrentDifficulty]    =
useState('Новачок'); // Default difficulty for new games
      const     [gameCompleteTime,     setGameCompleteTime]     =
useState(null);
      const [gameDifficulty, setGameDifficulty] = useState(null);
      const      [currentGameType,      setCurrentGameType]      =
useState(null); // 'mathecross', 'kenken', 'kakuro'

      const [userId, setUserId] = useState(null);
      const [isAuthReady, setIsAuthReady] = useState(false);

      // Firebase Auth initialization
      useEffect(() => {
        if    (!firebaseApp)    {    console.error("Firebase    не
ініціалізовано."); return; }
        const unsubscribe = onAuthStateChanged(auth, async (user)
=> {
          if (user) { setUserId(user.uid); } else {
            try {
              if        (initialAuthToken)        {        await
signInWithCustomToken(auth,  initialAuthToken);  }  else  {  await
signInAnonymously(auth); }
              setUserId(auth.currentUser.uid);
            } catch  (error)  {  console.error("Помилка анонімної
автентифікації:", error); setUserId(crypto.randomUUID()); }
          }
          setIsAuthReady(true);
        });
        return () => unsubscribe();
      }, []);

      // Save game result to Firestore
      useEffect(() => {
        if (gameCompleteTime !== null && gameDifficulty !== null
&& currentGameType !== null && userId && db) {
          const saveGameResult = async () => {
            try {
```

```
                const profileCollectionName = `${currentGameType}_profile`;
                const leaderboardCollectionName = `${currentGameType}_leaderboard`;

                // Update user's best time in their profile
                const userProfileRef = doc(db, `artifacts/${appId}/users/${userId}/${profileCollectionName}`, 'stats');
                const docSnap = await getDoc(userProfileRef);
                let currentBestTimes = {};
                if (docSnap.exists()) { currentBestTimes = docSnap.data().bestTimes || {}; }
                if (!currentBestTimes[gameDifficulty] || gameCompleteTime < currentBestTimes[gameDifficulty]) {
                  currentBestTimes[gameDifficulty] = gameCompleteTime;
                  await setDoc(userProfileRef, { bestTimes: currentBestTimes }, { merge: true });
                  console.log(`Оновлено найкращий час для профілю ${currentGameType}.`);
                }

                // Update global leaderboard
                const leaderboardCollectionRef = collection(db, `artifacts/${appId}/public/data/${leaderboardCollectionName}`);
                const querySnapshot = await getDocs(leaderboardCollectionRef);
                const existingEntries = querySnapshot.docs
                  .map(doc => ({ id: doc.id, ...doc.data() }))
                  .filter(entry => entry.difficulty === gameDifficulty && entry.userId === userId);

                if (existingEntries.length > 0) {
                  const userEntry = existingEntries[0];
                  if (gameCompleteTime < userEntry.time) {
                    await updateDoc(doc(db, `artifacts/${appId}/public/data/${leaderboardCollectionName}`, userEntry.id), {
                      time: gameCompleteTime,
                      timestamp: Date.now()
                    });
                    console.log(`Оновлено запис в рейтингу ${currentGameType}.`);
                  }
                } else {
                  await addDoc(leaderboardCollectionRef, {
                    userId: userId,
                    difficulty: gameDifficulty,
                    time: gameCompleteTime,
                    timestamp: Date.now()
                  });
```

```
                console.log(`Додано новий запис до рейтингу
${currentGameType}.`);
              }
            } catch (error) { console.error("Помилка збереження
результату гри:", error); }
            finally {
              setGameCompleteTime(null);
              setGameDifficulty(null);
              setCurrentGameType(null);
            }
          };
          saveGameResult();
        }
      }, [gameCompleteTime, gameDifficulty, currentGameType,
userId, db]);


      const toggleTheme = () => {
        setTheme((prevTheme) => (prevTheme === 'light' ? 'dark'
: 'light'));
      };

      useEffect(() => {
        document.documentElement.classList.remove('light',
'dark');
        document.documentElement.classList.add(theme);
      }, [theme]);

      const renderView = () => {
        if (!isAuthReady) {
          return (
            <div className="flex justify-center items-center h-
screen bg-gray-100 dark:bg-gray-900 text-gray-800 dark:text-gray-
200">
              <p className="text-xl">Ініціалізація Firebase та
автентифікація...</p>
            </div>
          );
        }

        switch (currentView) {
          case 'mainMenu':
            return (
              <div className="flex flex-col items-center justify-
center min-h-screen bg-gray-100 dark:bg-gray-900 text-gray-800
dark:text-gray-200 p-4">
                <h1 className="text-5xl font-extrabold text-
blue-700 dark:text-blue-300 mb-10 text-center">
                  Математичні Головоломки
                </h1>
```

```
                <div className="w-full max-w-sm bg-white dark:bg-
gray-800 p-8 rounded-2xl shadow-xl flex flex-col items-center
space-y-5">
                  <button
                    onClick={() => setCurrentView('gameSelect')}
                    className="w-full px-6 py-3 bg-blue-600
hover:bg-blue-700 text-white font-semibold rounded-lg shadow-md
transition-all duration-300 transform hover:scale-105"
                  >
                    Нова гра
                  </button>
                  <button
                    onClick={()                                 =>
setCurrentView('profileSelect')}
                    className="w-full px-6 py-3 bg-indigo-600
hover:bg-indigo-700 text-white font-semibold rounded-lg shadow-md
transition-all duration-300 transform hover:scale-105"
                  >
                    Профіль
                  </button>
                  <button
                    onClick={()                                 =>
setCurrentView('leaderboardSelect')}
                    className="w-full px-6 py-3 bg-green-600
hover:bg-green-700 text-white font-semibold rounded-lg shadow-md
transition-all duration-300 transform hover:scale-105"
                  >
                    Рейтинг
                  </button>
                  <button
                    onClick={() => setCurrentView('helpSelect')}
                    className="w-full px-6 py-3 bg-yellow-600
hover:bg-yellow-700 text-white font-semibold rounded-lg shadow-md
transition-all duration-300 transform hover:scale-105"
                  >
                    Довідка
                  </button>
                  <button
                    onClick={() => { /* In a web app, 'Exit' might
clear    state    or    simply    return    to    menu.    */
setCurrentView('mainMenu'); }}
                    className="w-full px-6 py-3 bg-red-600
hover:bg-red-700 text-white font-semibold rounded-lg shadow-md
transition-all duration-300 transform hover:scale-105"
                  >
                    Вихід
                  </button>
                </div>

                <button
                  onClick={toggleTheme}
```

```
                  className="mt-8 px-6 py-2 bg-gray-300 dark:bg-
gray-700 text-gray-800 dark:text-gray-200 rounded-full shadow-lg
transition-colors duration-300"
                >
                  Перемкнути на {theme === 'light' ? 'Темну' :
'Світлу'} тему
                </button>
              </div>
            );
          case 'gameSelect':
            return (
              <div className="flex flex-col items-center justify-
center min-h-screen bg-gray-100 dark:bg-gray-900 text-gray-800
dark:text-gray-200 p-4">
                <h2 className="text-4xl font-bold text-blue-600
dark:text-blue-400 mb-8">Виберіть головоломку</h2>
                <div className="w-full max-w-sm bg-white dark:bg-
gray-800 p-8 rounded-2xl shadow-xl flex flex-col items-center
space-y-5">
                  <label             htmlFor="difficulty-select"
className="block text-lg font-semibold text-gray-700 dark:text-
gray-300 mb-2">
                    Виберіть складність:
                  </label>
                  <select
                    id="difficulty-select"
                    value={currentDifficulty}
                    onChange={(e)                                =>
setCurrentDifficulty(e.target.value)}
                    className="w-full p-3 border border-gray-300
dark:border-gray-600 rounded-lg bg-gray-50 dark:bg-gray-700 text-
gray-900  dark:text-gray-100  focus:ring-blue-500  focus:border-
blue-500"
                  >
                    <option value="Новачок">Новачок</option>
                    <option
value="Досвідчений">Досвідчений</option>
                    <option
value="Професіонал">Професіонал</option>
                    <option value="Експерт">Експерт</option>
                    <option value="Геній">Геній</option>
                  </select>
                  <button
                    onClick={()                =>                {
setCurrentView('mathecrossGame');
setCurrentGameType('mathecross'); }}
                    className="w-full  px-6  py-3  bg-emerald-600
hover:bg-emerald-700 text-white font-semibold rounded-lg shadow-
md transition-all duration-300 transform hover:scale-105"
                  >
                    Математичний Кросворд
                  </button>
                  <button
```

```
                    onClick={()                    =>                    {
setCurrentView('kenkenGame'); setCurrentGameType('kenken'); }}
                    className="w-full  px-6  py-3  bg-cyan-600
hover:bg-cyan-700 text-white font-semibold rounded-lg shadow-md
transition-all duration-300 transform hover:scale-105"
                  >
                    КенКен
                  </button>
                  <button
                    onClick={()                    =>                    {
setCurrentView('kakuroGame'); setCurrentGameType('kakuro'); }}
                    className="w-full  px-6  py-3  bg-orange-600
hover:bg-orange-700 text-white font-semibold rounded-lg shadow-md
transition-all duration-300 transform hover:scale-105"
                  >
                    Кауро
                  </button>
                </div>
                <button
                  onClick={() => setCurrentView('mainMenu')}
                  className="mt-8 px-8 py-3 bg-gray-500 hover:bg-
gray-600 text-white rounded-lg shadow-lg transition-colors"
                >
                  Повернутися до головного меню
                </button>
              </div>
            );
          case 'profileSelect':
            return (
              <div className="flex flex-col items-center justify-
center  min-h-screen  bg-gray-100  dark:bg-gray-900  text-gray-800
dark:text-gray-200 p-4">
                <h2 className="text-4xl font-bold text-blue-600
dark:text-blue-400 mb-8">Виберіть профіль гри</h2>
                <div className="w-full max-w-sm bg-white dark:bg-
gray-800  p-8  rounded-2xl  shadow-xl  flex  flex-col  items-center
space-y-5">
                  <button
                    onClick={()                    =>                    {
setCurrentView('mathecrossProfile'); }}
                    className="w-full  px-6  py-3  bg-emerald-600
hover:bg-emerald-700 text-white font-semibold rounded-lg shadow-
md transition-all duration-300 transform hover:scale-105"
                  >
                    Профіль Математичного Кросворду
                  </button>
                  <button
                    onClick={()                    =>                    {
setCurrentView('kenkenProfile'); }}
                    className="w-full  px-6  py-3  bg-cyan-600
hover:bg-cyan-700 text-white font-semibold rounded-lg shadow-md
transition-all duration-300 transform hover:scale-105"
                  >
```

```
                  Профіль КенКен
                </button>
                <button
                  onClick={()                =>                {
setCurrentView('kakuroProfile'); }}
                  className="w-full  px-6  py-3  bg-orange-600
hover:bg-orange-700 text-white font-semibold rounded-lg shadow-md
transition-all duration-300 transform hover:scale-105"
                >
                  Профіль Кауро
                </button>
              </div>
              <button
                onClick={() => setCurrentView('mainMenu')}
                className="mt-8 px-8 py-3 bg-gray-500 hover:bg-
gray-600 text-white rounded-lg shadow-lg transition-colors"
              >
                Повернутися до головного меню
              </button>
            </div>
          );
        case 'leaderboardSelect':
          return (
            <div className="flex flex-col items-center justify-
center  min-h-screen  bg-gray-100  dark:bg-gray-900  text-gray-800
dark:text-gray-200 p-4">
              <h2 className="text-4xl font-bold text-blue-600
dark:text-blue-400 mb-8">Виберіть рейтинг гри</h2>
              <div className="w-full max-w-sm bg-white dark:bg-
gray-800  p-8  rounded-2xl  shadow-xl  flex  flex-col  items-center
space-y-5">
                <button
                  onClick={()                =>                {
setCurrentView('mathecrossLeaderboard'); }}
                  className="w-full  px-6  py-3  bg-emerald-600
hover:bg-emerald-700 text-white font-semibold rounded-lg shadow-
md transition-all duration-300 transform hover:scale-105"
                >
                  Рейтинг Математичного Кросворду
                </button>
                <button
                  onClick={()                =>                {
setCurrentView('kenkenLeaderboard'); }}
                  className="w-full   px-6   py-3   bg-cyan-600
hover:bg-cyan-700 text-white font-semibold rounded-lg shadow-md
transition-all duration-300 transform hover:scale-105"
                >
                  Рейтинг КенКен
                </button>
                <button
                  onClick={()                =>                {
setCurrentView('kakuroLeaderboard'); }}
```

```
                    className="w-full  px-6  py-3  bg-orange-600
hover:bg-orange-700 text-white font-semibold rounded-lg shadow-md
transition-all duration-300 transform hover:scale-105"
                  >
                    Рейтинг Кауро
                  </button>
                </div>
                <button
                  onClick={() => setCurrentView('mainMenu')}
                  className="mt-8 px-8 py-3 bg-gray-500 hover:bg-
gray-600 text-white rounded-lg shadow-lg transition-colors"
                >
                  Повернутися до головного меню
                </button>
              </div>
            );
          case 'helpSelect':
            return (
              <div className="flex flex-col items-center justify-
center  min-h-screen  bg-gray-100  dark:bg-gray-900  text-gray-800
dark:text-gray-200 p-4">
                <h2 className="text-4xl font-bold text-blue-600
dark:text-blue-400 mb-8">Виберіть довідку гри</h2>
                <div className="w-full max-w-sm bg-white dark:bg-
gray-800  p-8  rounded-2xl  shadow-xl  flex  flex-col  items-center
space-y-5">
                  <button
                    onClick={()               =>               {
setCurrentView('mathecrossHelp'); }}
                    className="w-full  px-6  py-3  bg-emerald-600
hover:bg-emerald-700 text-white font-semibold rounded-lg shadow-
md transition-all duration-300 transform hover:scale-105"
                  >
                    Довідка Математичного Кросворду
                  </button>
                  <button
                    onClick={()               =>               {
setCurrentView('kenkenHelp'); }}
                    className="w-full   px-6   py-3   bg-cyan-600
hover:bg-cyan-700  text-white  font-semibold  rounded-lg  shadow-md
transition-all duration-300 transform hover:scale-105"
                  >
                    Довідка КенКен
                  </button>
                  <button
                    onClick={()               =>               {
setCurrentView('kakuroHelp'); }}
                    className="w-full  px-6  py-3  bg-orange-600
hover:bg-orange-700 text-white font-semibold rounded-lg shadow-md
transition-all duration-300 transform hover:scale-105"
                  >
                    Довідка Кауро
                  </button>
```

```
                </div>
                <button
                  onClick={() => setCurrentView('mainMenu')}
                  className="mt-8 px-8 py-3 bg-gray-500 hover:bg-
gray-600 text-white rounded-lg shadow-lg transition-colors"
                >
                  Повернутися до головного меню
                </button>
              </div>
            );
          case 'mathecrossGame':
            return    <MathecrossGame    db={db}    auth={auth}
userId={userId}                              setView={setCurrentView}
difficulty={currentDifficulty}
setGameCompleteTime={setGameCompleteTime}
setGameDifficulty={setGameDifficulty}
setCurrentGameType={setCurrentGameType} />;
          case 'kenkenGame':
            return     <KenKenGame     db={db}     auth={auth}
userId={userId}                              setView={setCurrentView}
difficulty={currentDifficulty}
setGameCompleteTime={setGameCompleteTime}
setGameDifficulty={setGameDifficulty}
setCurrentGameType={setCurrentGameType} />;
          case 'kakuroGame':
            return     <KakuroGame     db={db}     auth={auth}
userId={userId}                              setView={setCurrentView}
difficulty={currentDifficulty}
setGameCompleteTime={setGameCompleteTime}
setGameDifficulty={setGameDifficulty}
setCurrentGameType={setCurrentGameType} />;
          case 'mathecrossProfile':
            return  <MathecrossProfile  db={db}  userId={userId}
setView={setCurrentView} />;
          case 'kenkenProfile':
            return   <KenKenProfile   db={db}   userId={userId}
setView={setCurrentView} />;
          case 'kakuroProfile':
            return   <KakuroProfile   db={db}   userId={userId}
setView={setCurrentView} />;
          case 'mathecrossLeaderboard':
            return        <MathecrossLeaderboard        db={db}
setView={setCurrentView} />;
          case 'kenkenLeaderboard':
            return          <KenKenLeaderboard          db={db}
setView={setCurrentView} />;
          case 'kakuroLeaderboard':
            return          <KakuroLeaderboard          db={db}
setView={setCurrentView} />;
          case 'mathecrossHelp':
            return <MathecrossHelp setView={setCurrentView} />;
          case 'kenkenHelp':
            return <KenKenHelp setView={setCurrentView} />;
```

```
            case 'kakuroHelp':
              return <KakuroHelp setView={setCurrentView} />;
            default:
              return null;
          }
        };

        return (
          <div  className={`min-h-screen  font-sans  antialiased
${theme === 'dark' ? 'dark bg-gray-900 text-gray-100' : 'bg-gray-
100 text-gray-900'}`}>
            <style>{`
              @import
url('https://fonts.googleapis.com/css2?family=Inter:wght@400;600
;700&display=swap');
              body { font-family: 'Inter', sans-serif; }
              input[type="number"]::-webkit-inner-spin-button,
              input[type="number"]::-webkit-outer-spin-button {
                -webkit-appearance: none;
                margin: 0;
              }
              input[type="number"] {
                -moz-appearance: textfield;
              }
            `}</style>
            {renderView()}
          </div>
        );
      };

      export default App;
```