Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота

на тему: «Розробка методу та програмних засобів реалізації вебсистеми для розв'язування логічних задач»

Виконав: студент IV курсу
групи 4ПІ-21б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Топольніцький Т. В.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. ПЗ Войтко В. В.
(прізвище та ініціали)

Рецензент: к.т.н., доцент каф. ОТ Городецька О. С.
(прізвище та ініціали)

Допущено до захисту

Зав. кафедри ПЗ Романюк О.Н.
«09» 06 2025 р.

ВНТУ – 2025

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший бакалаврський
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
«21» березня 2025 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Топольніцькому Тарасу Віталійовичу

1. Тема роботи – «Розробка методу та програмних засобів реалізації вебсистеми для розв'язування логічних».

Керівник роботи: Войтко Вікторія Володимирівна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 20 березня 2025 р. №97.

2. Термін подання студентом роботи: 2 червня 2025 року.

3. Вихідні дані до роботи: середовище розробки – Visual Studio Code, система зберігання даних – PHP + JSON-файл; мови розробки – HTML, CSS, JavaScript, PHP; операційна система – Windows 8/10/11.

4. Зміст текстової частини: вступ, аналіз стану розвитку вебсистем для розв'язування логічних задач та постановка задач розробки, розробка методу, моделі та алгоритмів роботи програми, розробка програмних засобів для реалізації вебсистеми, тестування вебсистеми, висновки, список використаних джерел, додатки, ілюстративна частина.

5. Перелік ілюстративного матеріалу: титульний слайд – автор, науковий керівник бакалаврської кваліфікаційної роботи, тема; мета, об'єкт та предмет дослідження; задачі дослідження; наукова новизна; практична цінність отриманих результатів; блок-схема загального алгоритму роботи вебсистеми; блок-схема алгоритму методу формування рейтингу; модель роботи системи; блок-схема алгоритму генерування числа; uml-діаграми; схематичний дизайни

інтерфейсу; логування сервера; функціонал розробленої вебсистеми; апробація та публікації результатів роботи; впровадження; фінальний слайд.

6. Консультанти розділів бакалаврської кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1 – 4	Войтко В. В., к.т.н., доцент кафедри ПЗ	24.03.25	01.06.25

7. Дата видачі завдання 24 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз розвитку вебсистем розв'язування логічних задач та постановка задачі	25.03.25 – 01.04.2025	Виконано
2	Розробка методу, моделі та алгоритмів роботи вебсистеми	01.04.25 – 01.05.2025	Виконано
3	Розробка програмних модулів вебсистеми для розв'язування логічних задач	01.05.25 – 10.05.2025	Виконано
4	Тестування програми	10.05.25 – 20.05.2025	Виконано
5	Оформлення матеріалів до захисту БКР	20.05.2025 – 30.05.25	Виконано

Студент Топольніцький Т. В.
(підпис) (прізвище та ініціали)

Керівник бакалаврської кваліфікаційної роботи Войтко В. В.

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота містить 79 сторінок формату А4, на яких є 45 рисунків і 3 таблиць, а список використаних джерел складається з 25 найменувань.

У бакалаврській кваліфікаційній роботі проведено аналіз розвитку вебсистем розв'язування логічних задач та постановка задачі, що допомагає в полегшенні знаходження відповідних логічних завдань в інтернеті та тренерування мислення. Було проведено аналіз аналогів, визначено їх переваги і недоліки та доведено доцільність розробки власного вебсервісу.

Розроблено вебсистему, що має зручний та інтуїтивно зрозумілий інтерфейс для взаємодії користувачів із вебсистемою під час розв'язання логічних задач. Система має функції додавання та видалення нових задач, що базуються в адміністративній панелі, яка реалізує єдиний стандарт до коду логічних задач та дозволить керувати кодом розроблених задач. Система формує рейтинг до завдань.

Вебсистему було реалізовано з допомогою сучасних вебтехнологій, таких як HTML, CSS і JavaScript для розробки клієнтської частини та PHP для реалізації серверної логіки. Visual Studio Code було використано як основне середовище для розробки.

У результаті виконання бакалаврської кваліфікаційної роботи отримано вебсистему, що підвищить ефективність пошуку логічних задач для пересічного користувача інтернет.

Отримані в бакалаврській кваліфікаційній роботі результати можна використати для покращення наявних систем рейтингу та покращення масштабування вебсистем розв'язання логічних задач.

Ключові слова: вебсистеми, розв'язання логічних задач, система рейтингу.

ABSTRACT

The bachelor's thesis consists of 79 A4 pages, with 45 figures and 3 tables, and the list of references includes 25 titles.

The bachelor's thesis analyzes the development of web-based logic problem solving systems and formulates a problem that helps to facilitate the search for suitable logic problems on the Internet and training of thinking. The analysis of analogs was carried out, their advantages and disadvantages were determined, and the expediency of developing our own web service was proved.

A web system has been developed that allows to have a convenient and intuitive interface for user interaction with the web system while solving logical problems. To have the functions of adding and deleting new tasks, which will be based in the administrative panel, which implements a single standard for the code of logical tasks and will allow you to manage the code of developed tasks. Have a rating for tasks

The web system was implemented in a programming language using modern web technologies such as HTML, CSS and pure JavaScript for the development of the client side and PHP for the implementation of server logic. Visual Studio Code was used as the main development environment

As a result of the bachelor's thesis, a web system was developed that will increase the efficiency of searching for logical problems for an average Internet user.

The results obtained in the bachelor's thesis can be used to improve existing rating systems and improve the scalability of web-based logic problem solving systems.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. У сучасному суспільстві, яке характеризується активним розвитком технологій, зростає інтерес до інтелектуальних задач як інструментів когнітивного тренування, саморозвитку й ефективного дозвілля [1]. Логічні задачі посідають особливе місце серед таких засобів, оскільки сприяють формуванню аналітичного мислення [2], розвитку пам’яті та здатності до комплексного розв’язання проблем [3].

Разом із тим, аналіз існуючих платформ засвідчує фрагментарність та низький ступінь якості цифрових ресурсів, призначених для розв’язування логічних задач. Більшість із них орієнтовані на один тип завдань, доступні лише з окремих пристроїв або не зберігають статистику розв’язань, що ускладнює інтерактивність і систематичне використання таких рішень у когнітивних практиках. Враховуючи вказане, актуальним є створення спеціалізованої вебсистеми, яка б об’єднала логічні задачі різного типу, забезпечила зручний доступ з будь-яких пристроїв, фіксацію часу розв’язання, рейтингування користувачів та адаптивну складність завдань.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою бакалаврської кваліфікаційної роботи є покращення для користувачів режиму тренування логічного мислення шляхом розробки і використання спеціалізованої вебсистеми, яка забезпечує реалізацію механізму формування і перевірки задач визначених типів, фіксацію часу розв’язання задач, реалізує формуваня рейтингу користувачів і забезпечує можливість доступу з різних типів пристроїв, що дозволяє реалізувати зручний вебпортал для тренування і розвитку логічного мислення користувача.

Відповідно до поставленої мети потрібно вирішити такі завдання:

- провести аналіз існуючих програмних рішень для розв’язування логічних задач та визначити їхні переваги й недоліки;
- визначити функціональні вимоги до системи та розробити алгоритм взаємодії користувача з платформою;
- розробити модуль генерації логічних задач типу «Бики і корови» та «Задача Ейнштейна»;
- розробити режим адміністрування, який дозволяє додавати, редагувати, видаляти логічні задачі в системі;
- розробити централізовану систему оновлення та збереження даних;
- реалізувати механізм збереження результатів, формування рейтингу користувачів та фіксації часу отримання результату;
- побудувати інтерфейс вебсистеми з урахуванням адаптивності та сучасних принципів UI/UX-дизайну;
- провести тестування програмного продукту та сформувати рекомендації з його використання.

Об’єктом дослідження виступають процеси розробки та впровадження вебсистеми для організації вирішення логічних задач у вебсередовищі.

Предметом дослідження виступають методи та засоби реалізації вебсистеми для інтерактивного розв’язання логічних задач.

Методи дослідження. У процесі дослідження використовувалися такі методи:

– методи і технології модульної розробки вебсистем для програмної реалізації модулів системи;

– методи візуалізації даних результатів звітності для забезпечення отримання інформативної аналітики роботи вебсистеми;

– методи UI/UX для створення та перевірки інтерфейсів вебсистеми;

– методи тестування для перевірки працездатності роботи програми.

Новизна отриманих результатів:

1. Подальшого розвитку отримав метод формування рейтингів і зберігання часу проходження ігор, який, на відміну від існуючих підходів, забезпечує інтеграцію рейтингової системи безпосередньо в код ігор і централізоване управління даними через JSON-структуру, дозволяючи динамічно оновлювати списки ігор і зберігати результати гравців у реальному часі, що дозволяє користувачам вебсистеми зручно переглядати актуальні рейтинги, порівнювати свої результати з іншими гравцями, швидко отримувати доступ до нових ігор без необхідності оновлення клієнтської частини.

2. Подальшого розвитку отримала модель вебсистеми для розв’язування логічних задач, яка, на відміну від існуючих, зручно поєднує просте управління іграми через JSON-файли та зрозумілу адміністративну панель, забезпечує чіткий формат збору даних про рейтинги й час проходження безпосередньо в коді ігор, а також пропонує зручний інтерфейс для гравців і адміністраторів, що дозволяє підвищити зручність використання розробленої вебсистеми та сприяє залученню широкого кола користувачів.

Практична цінність отриманих результатів полягає в можливості використанні розроблених програмних модулів системи для розв’язування логічних задач, завдяким яким користувачі зможуть комфортно та цікаво проводити час за улюбленим заняттям, а також розривати логічне мислення. Програмні модулі можуть бути використані розробниками для створення аналогічних систем у галузях для розв’язання задач включно з випадкової генерації задачі «Бики та Корови», та «Задачі Енштейна». Також реалізовано програмний модуль для додання адміністратором нового типу задач з логіки за допомогою адміністративної панелі без необхідності змінювати код на серверній частині, що дозволяє отримувати результат в реальному часі використання вебсистеми. Таке програмне забезпечення може бути адаптоване для використання в освітніх установах для кращого засвоєння

матеріалу з логіки та відпрацювання розв’язання різноманітного типу логічних задач.

Особистий внесок здобувача. Усі результати, подані в бакалаврській кваліфікаційній роботі, були отримані автором особисто. У науковій праці [4], опублікованій у співавторстві, автору належать такі результати: таблиця порівняння функціональних характеристик аналогів, та програмна реалізація розроблених функцій.

Апробація та публікація результатів проєкту. Результати бакалаврської кваліфікаційної роботи обговорювались на Всеукраїнській науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії (2025) і опубліковані в тезах доповідей цієї конференції [4].

Аналіз. У пояснювальній записці до бакалаврської кваліфікаційної роботи було розглянуто чотири розділи та було використано 25 літературних джерел.

1 АНАЛІЗ РОЗВИТКУ ВЕБСИСТЕМ РОЗВ'ЯЗУВАННЯ ЛОГІЧНИХ ЗАДАЧ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз стану вебсистем розв'язання логічних задач

На сучасному етапі розвитку інформаційних технологій вебсистеми стають важливою складовою різних аспектів діяльності людини, таких як навчання, розваги та розвиток інтелектуальних здібностей. Основною перевагою цих систем є можливість взаємодії користувачів з інформацією у режимі реального часу, що забезпечує доступність ресурсів незалежно від місця розташування користувача. Завдяки вебтехнологіям користувачі можуть ефективно опановувати нові знання, розвивати навички логічного мислення, вирішуючи спеціалізовані задачі та головоломки онлайн.

Сучасні веборієнтовані інформаційні системи слугують універсальними платформами, що забезпечують формування просторово масштабованих аналітичних звітів, автоматизований облік гуманітарної допомоги та виконання інших задач оброблення даних [5,6].

Вебсистема – це інтегрована клієнт-серверна інформаційна система, що працює поверх мережевих протоколів, використовує браузер як універсальний інтерфейс і характеризується розподіленою архітектурою, гнучкою масштабованістю та можливістю безперервного оновлення функціональності в режимі реального часу. Сучасні веборієнтований інформаційні системи слугують універсальними платформами для формування просторово масштабованих аналітичних звітів, автоматизованого обліку гуманітарної допомоги та виконання інших задач оброблення даних [5,6]. Вебсистема, розроблена у цьому дослідженні, призначена для надання користувачам мережі Інтернет доступу до тренувань з логічних задач.

Вебсистеми характеризуються високою гнучкістю та доступністю, оскільки можуть бути використані на будь-яких пристроях з підключенням до інтернету, включаючи мобільні телефони, планшети та персональні комп'ютери. Це дозволяє значно розширити аудиторію, яка має змогу

користуватися такими ресурсами, незалежно від віку, освіти або технічних навичок [7].

Інтерактивність є ключовою характеристикою вебсистем, яка робить їх привабливими для користувачів. Вона проявляється у можливості негайної реакції системи на дії користувача, наданні зворотного зв’язку, підказок та рекомендацій у процесі розв'язання завдань. Це суттєво підвищує мотивацію користувачів та дозволяє ефективніше досягати освітніх та розважальних цілей.

Серед вебсистем для логічних ігор та головоломок можна виокремити як вузькоспеціалізовані ресурси, що зосереджені на одному типі завдань, так і багатофункціональні платформи, які пропонують широкий вибір задач різного рівня складності. Перший тип платформ характеризується глибокою оптимізацією та зручністю роботи з певною категорією задач, тоді як другий – пропонує різноманітність завдань та інтегровані можливості соціальної взаємодії користувачів.

Розвиток сучасних вебсистем передбачає постійне вдосконалення користувацького досвіду, оновлення контенту та впровадження інноваційних технологій, що дозволяє задовольнити зростаючі потреби користувачів у якісних освітніх і розважальних ресурсах. Вебсистеми, які поєднують інтерактивність, гейміфікацію та структурований підхід до розвитку логічних навичок, мають значний потенціал для подальшого зростання популярності серед широкого кола користувачів [7].

Логічні задачі [8] є важливою частиною інтелектуального розвитку людини, яка покликана стимулювати роботу мозку та покращувати когнітивні функції. Їхнє вирішення сприяє розвитку не лише логічного мислення, а й посилює концентрацію уваги, пам’ять та здатність до дедуктивних міркувань. Виконання таких завдань дозволяє індивіду ефективно аналізувати складні ситуації, знаходити оптимальні рішення в повсякденному житті та професійній діяльності.

До поширених типів логічних задач [8] належать числові задачі, які ґрунтуються на операціях з числами і математичних залежностях. Їхнє розв’язання передбачає використання арифметичних та алгебраїчних підходів, що формує у користувача навички швидких розрахунків та числової логіки. Завдяки таким задачам можна покращити математичну грамотність та аналітичні здібності.

Графічні логічні задачі [8] вирішуються за допомогою візуальної інформації та графічних моделей. Вони потребують від користувача високого рівня просторового мислення та візуалізації. Популярними прикладами таких задач є пазли, лабіринти, графічні послідовності та задачі з просторовими головоломками, що активно використовуються у тренуванні просторового та конструктивного мислення.

Словесні логічні задачі [8] включають завдання на розуміння тексту, аналіз смислових зв’язків та пошук логічних суперечностей або підтверджень у наведених твердженнях. Їхнє вирішення розвиває у користувачів вербальні навички, здатність чітко формулювати свої думки, а також критично оцінювати отриману інформацію. Такі задачі часто використовуються для підготовки до інтелектуальних змагань, дебатів або навчання аналітичному читанню.

Комбінаторні логічні задачі [8] передбачають застосування теорії ймовірностей, перестановок, сполучень та інших математичних методів для знаходження розв’язку. Вони особливо важливі для розвитку стратегічного мислення, оскільки вимагають передбачення результатів багатьох можливих сценаріїв і оптимального вибору серед них. Такі задачі використовуються у програмуванні, економічних прогнозах та інших сферах, де потрібно приймати ефективні рішення в умовах невизначеності.

Задача «Бики та корови» є популярною логічною грою, яка належить до класу дедуктивних задач. Основним завданням гравця є відгадати приховане число, використовуючи підказки, що отримуються після кожного припущення. Важливим аспектом цієї гри є наявність двох типів підказок:

кількість цифр, що є правильними, але стоять на неправильних позиціях («корови»), і кількість цифр, які є правильними і стоять на правильних місцях («бики»).

Гра розпочинається з того, що один гравець (або комп'ютер) загадує число, зазвичай із 4 або більше цифр. Всі цифри повинні бути унікальними. Інший гравець намагається вгадати це число, пропонуючи різні варіанти. Після кожної спроби він отримує підказку про кількість «биків» та «корів», що дозволяє методом виключення наближатися до правильного рішення [9].

Особливість гри полягає у тому, що вона вимагає активного застосування логічного мислення та дедуктивного підходу. Гравець має систематично аналізувати отримані підказки, виключати неможливі варіанти та будувати стратегію наступних припущень для мінімізації кількості спроб.

У зв’язку з цим гра «Бики та корови» є чудовим інструментом для розвитку навичок аналітичного мислення, логіки та концентрації уваги. Вона ефективно використовується як у розважальних, так і в освітніх цілях, зокрема для навчання дітей та дорослих дедуктивним прийомам мислення.

Ця гра може бути реалізована як у традиційному форматі з ручкою та папером, так і у цифрових форматах, включаючи мобільні додатки, вебсистеми та онлайн-ігри. Цифрові реалізації надають додаткові можливості, такі як автоматичний підрахунок спроб, підказки, ведення статистики і рейтингів.

Різноманітні рівні складності, які можуть бути реалізовані у цифрових версіях гри, дозволяють адаптувати задачу до різних вікових груп та рівнів підготовки користувачів. Простий рівень зазвичай включає меншу кількість цифр, тоді як складніші рівні можуть додавати більше цифр або скорочувати кількість спроб для досягнення рішення.

Також слід зазначити, що «Бики та корови» має велике значення для тренування оперативної пам'яті та здатності утримувати в пам'яті і порівнювати різні комбінації цифр, що позитивно впливає на загальний когнітивний розвиток.

Завдяки своїй простоті, зрозумілості правил і водночас складності логічних стратегій, гра користується популярністю серед широкого кола користувачів. Також використовується у змаганнях, інтелектуальних турнірах та навчальних програмах для розвитку логічного мислення та математичних здібностей.

Різноманітні варіанти реалізації гри дозволяють інтегрувати її у різні платформи, включаючи освітні системи, що підтримують навчання STEM-напрямкам. Вебсистеми, які пропонують гру «Бики та корови», можуть також використовуватись для аналізу поведінки користувачів і надавати персоналізовані рекомендації для розвитку їхніх когнітивних здібностей.

Таким чином, гра «Бики та корови» не лише залишається популярною формою інтелектуальної розваги, але й активно використовується як ефективний інструмент для тренування логічних та дедуктивних навичок, що є важливими для багатьох аспектів навчання та професійного життя.

Задача Ейнштейна, також відома як логічна грід-задача або загадка Зебри, є однією з найвідоміших логічних задач, що вимагає високого рівня логічного мислення, дедукції та аналітичних здібностей. Авторство цієї задачі часто приписують Альберту Ейнштейну, хоча історичних підтверджень цьому немає. Незважаючи на це, популярність та широке застосування цієї задачі для тренування логічних навичок є беззаперечними [10].

Суть задачі полягає в тому, що гравцю надається кілька категорій характеристик (наприклад, колір будинку, національність мешканця, вид напою, тварина та тип сигарет), які необхідно розподілити між кількома об'єктами або особами на основі низки умов. Всі умови представлені у вигляді логічних тверджень, які пов'язані між собою і дозволяють поступово виключати неможливі комбінації характеристик.

Використання таблиці або логічної решітки є типовим способом представлення інформації при розв'язанні цієї задачі. Кожна комірка в таблиці позначає можливу комбінацію характеристик, а процес вирішення полягає у

поступовому заповненні клітинок, використовуючи дедуктивний метод виключення і підтвердження.

Особливістю задачі Ейнштейна є те, що вона не має прямого математичного або арифметичного рішення, а розв’язання повністю залежить від здатності гравця логічно пов'язувати окремі факти та аналізувати умови, надані в завданні. Це робить її важливим засобом для тренування дедуктивних навичок та системного підходу до аналізу складних ситуацій.

Популярність задачі зумовлена не лише її цікавістю та інтелектуальним викликом, але й тим, що вона активно використовується в навчальних процесах, психологічних тестах та інтелектуальних змаганнях. Вона дозволяє оцінити рівень логічного мислення учасників та ефективність їхнього підходу до системного вирішення складних проблем.

Завдяки своїй складності, задача Ейнштейна активно застосовується у процесі навчання для розвитку критичного мислення, вміння структурувати інформацію, а також для покращення навичок стратегічного планування. Вона також є популярною формою інтелектуальних розваг і часто представлена у вигляді онлайн-ігор та мобільних додатків.

Реалізації цієї задачі в цифрових форматах мають численні переваги, серед яких інтерактивні підказки, автоматична перевірка відповідей. Такий формат значно розширює аудиторію та дозволяє гравцям ефективно розвивати свої логічні здібності.

Крім того, використання задачі Ейнштейна у вебсистемах надає можливість збору й аналізу статистичних даних щодо поведінки користувачів. Це може бути використано для створення персоналізованих навчальних програм та рекомендацій, які дозволяють більш ефективно розвивати логічні та аналітичні навички користувачів.

Завдяки своїй структурованості й глибокому логічному компоненту, задача Ейнштейна залишається одним із найбільш ефективних інструментів для розвитку складних когнітивних навичок і дедуктивного мислення у різних

вікових групах. Вона сприяє не лише інтелектуальному розвитку, але й формуванню здатності до критичного аналізу інформації.

Таким чином, задача Ейнштейна є важливим і популярним засобом для тренування логічного та системного мислення, а її цифрові реалізації відкривають додаткові можливості для інтерактивного навчання та розваг у сучасному інформаційному просторі.

1.2 Порівняльний аналіз аналогів

У сучасному світі, де цифрові технології відіграють важливу роль у повсякденному житті, все більше людей шукають цікаві та корисні способи проводити вільний час. Особливо популярними стають платформи, які допомагають розвивати логічне мислення, тренувати мозок і випробовувати свої інтелектуальні здібності. Існує багато цифрових ресурсів, що пропонують різні види головоломок та інтелектуальних ігор, які доступні на мобільних пристроях, у месенджерах та в онлайн-спільнотах. Розглянемо популярні ресурси як аналоги розроблюваної вебсистеми для розв'язування логічних задач.

Генератор головоломок Zebra – мобільний додаток для Android, який генерує унікальні логічні задачі з можливістю налаштування рівня складності. це гра, у якій потрібно складати головоломки за допомогою логічних міркувань. Zebra Puzzle Generator щоразу створює унікальні головоломки з індивідуальними рівнями складності [11].

Lastkatka Bot – це багатофункціональний телеграм бот в якому реалізована логічна гра «Бики і корови». Для доступу щоб вирішити логічні задачі типу «Бики і корови» обов’язково потрібний телеграм [12].

Puzzling Stack Exchange– це онлайн-спільнота для любителів головоломок. Сайт містить задачі різних категорій, включаючи криптографію, загадки з підказками, математичні головоломки, задачі на логіку та нестандартне мислення. Користувачі розв’язують логічні задачі. за правильні відповіді користувачі отримують бали та підвищують свою репутацію [13].

Для наочної демонстрації відмінностей розглянутих додатків було зведено їх переваги і недоліки у таблицю порівняння наведено у таблиці 1.1, де наявність певних характеристик помічена 1, а відсутність – 0.

Таблиця 1.1 – Порівняльний аналіз аналогів

Критерій порівняння	Генератор головоломок Zebra	Lastkatka Bot	Puzzling Stack	Власна розробка
Можливість розв’язувати логічні задачі типу «Задачі Ейнштейна»	1	0	0	1
Можливість розв’язувати логічні задачі типу «Бики і корови»	0	1	0	1
Можливостю додання нового типу логічних задач за допомогою адміністративної консолі	0	0	0	1
Доступність на всіх девайсах без особливих вимог	0	0	1	1
Збереження результатів часу розв’язування	0	1	0	1
Ведення рейтингу користувачів	0	1	1	1
Сумарний коефіцієнт	1	3	2	6

Аналізуючи таблицю 1.1, відзначимо, що власна розробка має вищий сумарний коефіцієнт за розглянутими критеріями у порівнянні з аналогами Генератор головоломок Zebra на 83,3% (100% - 1/6*100% = 83,3%), у порівнянні з телеграм ботом Lastkatka Bot на 50% (100% - 3/6*100% = 50%), у порівнянні з вебсистемою Puzzling Stack на 66,6% (100% - 2/6*100% = 66,6%). Враховуючи переваги й недоліки систем-аналогів, було визначено функціонал власної розробки вебсистеми для розв’язування логічних задач.

1.3 Аналіз методів розв'язання задачі

У межах реалізації вебсистеми для логічних головоломок порівнювалися кілька архітектурних підходів. Класична модель MVC забезпечує простоту розробки й централізований контроль безпеки, однак має недолік у вигляді повного перезавантаження сторінки після кожної взаємодії, що робить її менш придатною для динамічних ігор. Односторінкові застосунки (SPA) забезпечують миттєві переходи й плавну взаємодію користувача з системою, однак вимагають складнішого управління початковим завантаженням і оптимізацією SEO. Гібридна архітектура SPA + SSR надає можливість поєднати переваги обох підходів, проте значно ускладнює процес розгортання та потребує глибших технічних знань. Також розглядався варіант використання монолітної архітектури та мікросервісної моделі. Моноліт простий у розробці й підтримці, але обмежений у масштабуванні, тоді як мікросервіси забезпечують гнучкість і незалежність частин системи, проте вимагають складної інфраструктури та супроводу [14].

У розробленій вебсистемі було обрано монолітну архітектуру з використанням класичної моделі, де клієнтська частина реалізована на HTML, CSS і JavaScript, а серверна обробка даних здійснюється за допомогою PHP із збереженням інформації у JSON-файлах через REST API. Цей підхід виявився оптимальним через кілька причин. По-перше, він забезпечує простоту розгортання та обслуговування без потреби у складних серверних технологіях. По-друге, така архітектура є цілком достатньою для підтримки середнього навантаження, що очікується в системі навчального або демонстраційного типу. По-третє, продуктивність обраного рішення дозволяє швидко обробляти ігрові події без відчутних затримок для користувача. Окрім того, така система потребує мінімум ресурсів, що дає можливість працювати навіть на простих хостинг-платформах без баз даних чи додаткового програмного забезпечення. Простота розробки та супроводу робить цю архітектуру особливо привабливою для навчальних цілей, а легкість модифікацій – для подальшого розширення функціональності.

Таким чином, у межах вимог до системи для логічних задач, яка має обслуговувати невелике або середнє навантаження й бути зручною для розгортання та підтримки, обрана архітектурна модель є найбільш виправданою і ефективною.

Алгоритмічне ядро вебсистеми повинно коректно, швидко та масштабовано обробляти два різні класи головоломок: дедуктивну гру «Бики і корови» та логічну решітку «Загадка Ейнштейна». Також має існувати механізм впровадження нових задач що реалізовано за допомою стандартизації ключових атрибутів у коді для кожної поданої задачі. Кожен тип задачі ставить окремі вимоги до складності, структури даних і способів оптимізації, тому реалізацію організовано у вигляді полі-модульного Game Logic Engine.

Модель даних «Биків і корів». Секретне число з унікальних цифр зберігається як масив int[4]. Під час перевірки користувацької спроби Engine проходить масив одноразово, порівнюючи цифри за індексом (для «биків») і паралельно інкрементуючи частоти в byte[10] (для «корів»). Завдяки лінійній складності O(n) час перевірки залишається сталим навіть для довших секретів, що дозволяє підтримувати режим «швидкої гри» з обмеженням ≤ 100 мс на відповідь сервера.

Рейтингова система реалізована через стандартизацію елементів (rate, time, userName, gameLabel) у коді ігор, що перевіряються в adminPanel.html. Повна реалізація передбачає збереження результатів у ratings.json через saveRating.php і відображення в ratingViewer.html. Гравці відправляють результати асинхронно (fetch), а рейтинги відображаються динамічно. Система мотивує змагальність, підвищує залученість і спрощує керування.

Отже, на основі проведеного аналізу методів створення вебсистеми для розв’язання логічних задач було обрано оптимальні технологічні рішення, які забезпечують ефективність, надійність і простоту розробки та підтримки.

В якості серверної частини вебсистеми було обрано мову програмування PHP. Це рішення зумовлено кількома важливими факторами. По-перше, PHP

є однією з найбільш розповсюджених і підтримуваних мов для розробки вебдодатків, що дозволяє легко розгорнути систему на практично будь-якому хостингу без необхідності спеціальних налаштувань або встановлення додаткового програмного забезпечення. По-друге, PHP добре інтегрується з файловими системами, що дає змогу просто працювати із JSON-файлами для збереження даних, без потреби налаштовувати повноцінні бази даних, що ідеально підходить для невеликих або навчальних проєктів. Крім того, PHP має низький поріг входу і зрозумілий синтаксис, що спрощує розробку, налагодження і супровід системи. Важливо також, що PHP забезпечує достатню продуктивність для обробки запитів у режимі реального часу в системах із середнім навантаженням, таких як вебсистема логічних задач. Сукупність цих переваг робить PHP найбільш доцільним вибором для реалізації серверної логіки даного проєкту.

Для клієнтської частини обрано HTML та JavaScript, які забезпечують створення інтерактивного користувацького інтерфейсу без використання додаткових фреймворків. Завдяки цьому забезпечується контроль над структурою сторінки, динамічна взаємодія з користувачем, а також можливість оптимізації продуктивності та базової SEO-підтримки.

1.4 Постановка задачі

Після аналізу переваг та недоліків існуючих вебсистем було визначено завдання для розробки вебсистеми:

1. Розробити зручний та інтуїтивно зрозумілий інтерфейс для взаємодії користувачів із вебсистемою під час розв’язання логічних задач («Бики і корови» та «Задача Ейнштейна»).

2. Реалізувати функції додавання та видалення нових задач, яка буде базуватись в адміністративній панелі, яка реалізує єдиний стандарт до коду логічних задач та дозволить керувати кодом розроблених задач.

3. Спроєктувати модель вебсистеми, яка забезпечить зручну взаємодію користувача з інтерфейсом, рейтингом та статистикою, та буде підтримувати масштабування при додаванні та видаленні задач.

4. Розробити алгоритми роботи вебсистеми, які дозволять коректно відображати процес і результати вирішення задач, статистику користувачів та рейтинги та дозволяє додавати нові логічні задачі.

5. Реалізувати програмні модулі для керування процесом вирішення задач користувачем, збору статистики та формування рейтингу.

6. Провести тестування кожного з програмних модулів з метою забезпечення надійності й коректності роботи вебсистеми.

7. Розробити зрозумілу та детальну інструкцію для користувачів щодо роботи з вебсистемою.

Запропонована система повинна мати наступні переваги:

– Сучасний, простий та інтуїтивно зрозумілий інтерфейс (розроблений з використанням HTML та JavaScript), який забезпечить комфортну взаємодію користувача з логічними задачами;

– гнучка система налаштування складності задач, що враховує різний рівень підготовки користувачів;

– зручної та зрозумілої системи рейтингування та статистичного аналізу результатів для посилення мотивації користувачів;

– адаптивність та масштабованість системи, яка дозволить легко додавати нові задачі або розширювати вже наявні функції;

– надійність і швидкодія серверної частини на PHP із зберіганням інформації у форматі JSON-файлів.

Для виконання завдань було обрано такі методи і технології:

– клієнтська частина яка використовує HTML та JavaScript для створення адаптивного і зручного інтерфейсу;

– серверна частина яка використовує PHP для обробки запитів і організації роботи API;

– база даних яка використовує зберігання даних користувачів і статистики у форматі JSON-файлів для спрощення обробки та надійного збереження інформації без використання повноцінної реляційної бази даних;

– контроль якості має модульне та інтеграційне тестування програмних модулів.

1.5 Висновки

Було проаналізовано стан вебсистем розв’язання логічних задач.

Оглянуто Генератор головоломок Zebra, Lastkatka Bot, Puzzling Stack Exchange кожне з цих рішень має свої недоліки та переваги представленні у вигляді таблички 1.1. Проаналізовано сучасні методи розв’язання для поставленої задачі у розробці вебсистеми задачі. Проаналізовано існуючі архітектурні рішення та обгрунтоване рішення для розробки JSON-архітектури.

На основі проаналізованих аналогів та методів було поставлено задачу для роботи вебсистеми: створити зручний та інтуїтивно зрозумілий інтерфейс для взаємодії користувачів із системою під час розв’язання ігор «Бики і Корови» та «Задача Ейнштейна»; спроєктувати модель вебсистеми, яка забезпечить ефективну роботу з інтерфейсом, рейтингом та статистикою; розробити алгоритми, що дозволять коректно відображати процес вирішення задач, результати, а також статистику користувачів і рейтинги; реалізувати програмні модулі для керування процесом розв’язання задач, збору статистичних даних та формування рейтингів; провести тестування кожного з розроблених модулів для забезпечення надійності й стабільності роботи системи.

Таким чином було доведено доцільність розробки власної вебсистеми. На основі отриманої інформації було сформовано перелік задач, які необхідно виконати для розробки.

2 РОЗРОБКА МЕТОДУ, МОДЕЛІ ТА АЛГОРИТМУ РОБОТИ ВЕБСИСТЕМИ

2.1 Аналіз вхідних даних системи

Для реалізації вебсистеми для розв’язування логічних задач будуть використані поєднання мов програмування JavaScript та PhP.

JavaScript –це легка, інтерпретована мова програмування з функціями першого класу. Вона найбільш відома як мова сценаріїв для вебсторінок, але також використовується в багатьох середовищах поза браузером, таких як Node.js, Apache CouchDB та Adobe Acrobat. JavaScript є прототипно-орієнтованою, мультипарадигмовою, однопотоковою, динамічною мовою, яка підтримує об'єктно-орієнтоване, імперативне та декларативне (наприклад, функціональне програмування) стилі програмування [15].

PHP–це широко використовувана мова сценаріїв загального призначення з відкритим кодом, яка особливо підходить для веброзробки та може бути вбудована в HTML [16].

Поєднання JavaScript та PHP забезпечує повноцінну реалізацію вебсистеми для логічних задач: JavaScript відповідає за інтерактивність та динамічність на стороні клієнта, тоді як PHP обробляє серверні запити та керує даними.

На вході вебсистема для розв’язання логічних задач (зокрема – «Бики та Корови» та «Задача Ейнштейна») буде надходити набір структурованих даних, необхідних для початку, перебігу та завершення гри. Перш за все, користувач вводить своє ім’я, яке використовується для ідентифікації в системі та формування рейтингу. Далі обирається тип гри: «Бики та Корови» або «Задача Ейнштейна». У разі гри «Бики та Корови» користувач додатково встановлює довжину секретного числа (від 3 до 5 цифр), що впливає на складність гри. Протягом гри користувач вводить варіанти рішень – комбінації цифр або логічні структури залежно від типу гри. Кожна спроба фіксується в системі, а їх кількість обмежується згідно з обраними параметрами. Завершення гри

може бути автоматичним (у разі виграшу або програшу) або ініційованим користувачем вручну через кнопку «Здатися». Після завершення гри система фіксує час, дату, кількість спроб та обчислює оцінку (бал), яка залежить від складності гри та ефективності дій користувача. Усі ці дані формують єдиний об’єкт, що передається на сервер для збереження у форматі JSON. Таким чином, вхідні дані системи є контрольованими, валідованими та забезпечують точну фіксацію результатів для формування історії проходження та рейтингу.

Основною функцією вебсистеми є організація проходження логічних задач, зокрема «Бики та Корови» та «Задача Ейнштейна», із можливістю обліку спроб, часу виконання та підрахунку підсумкового балу для кожного користувача. Платформа забезпечує зручний інтерфейс для введення імені, вибору типу задачі та її параметрів, фіксацію кожної спроби вирішення та надання підказок у процесі гри. Окремо реалізовані функції автоматичного підбиття підсумків після завершення гри: система визначає результат (виграш або програш), розраховує оцінку та зберігає дані користувача.

Також у вебсистемі для розв’язування логічних задач, поділеній на модулі за функціональністю, кожен модуль має чітко визначені вхідні дані, необхідні для виконання своїх завдань.

Модуль адміністрування, який включає adminPanel.html і saveGame.php, приймає пароль адміністратора для автентифікації, а також дані гри (gameLabel і gamefile), введені через форму додавання, що надходять до saveGame.php через асинхронний fetch-запит у форматі JSON.

Модуль ігор, реалізований у Games/*.html, отримує вхідні дані з HTML-елементів гри, зокрема rate (бали), time (час проходження) і userName (ім’я гравця), які зчитуються JavaScript-кодом для подальшої відправки через fetch.

Модуль рейтингів, що охоплює ratingViewer.html і saveRating.php, використовує JSON-об’єкт із ratings.json для відображення рейтингів, а також приймає від saveRating.php надіслані дані гри через POST-запит.

Модуль даних, представлений params.json і ratings.json, не має прямих вхідних даних, але забезпечує доступ до метаданих ігор та рейтингів, які

читаються іншими модулями через HTTP-запити, формуючи основу для їхньої роботи

Завдяки такому підходу вебсистема демонструє гнучкість, масштабованість та зручність в експлуатації. Вона легко розширюється за рахунок додавання нових типів задач або вдосконалення існуючих механік, при цьому зберігаючи стабільність роботи та швидку реакцію на запити користувачів.

Отже, розроблена вебсистема забезпечує зручне проходження логічних задач, ефективний облік результатів та формування рейтингів, що сприяє підвищенню мотивації користувачів і розвитку їх аналітичного мислення, водночас залишаючись простою в розгортанні, надійною у роботі та легкою для подальшого розширення.

2.2 Розробка методу формування рейтингів і зберігання часу проходження ігор

Метою цього методу є створення рейтингової системи для вебсистеми розв’язування логічних задач, яка базується на обчисленні рейтингу гравців. Рейтинг формується на основі двох ключових показників: очок, що відображають успішність проходження гри, і часу, який вимірює швидкість виконання. Кожен запис у рейтингу асоціюється з іменем гравця, що дозволяє ідентифікувати досягнення та забезпечити змагальний аспект. Цей підхід мотивує користувачів покращувати свої результати та сприяє освітнім цілям системи.

Архітектура методу опирається на централізоване зберігання даних у файлі ratings.json, який слугує основним сховищем для рейтингів, структурованим за файлами ігор (наприклад, bullsAndCows.html) з масивами записів. Обробка даних здійснюється через серверний скрипт saveRating.php, який отримує результати від клієнтського коду, валідує їх і оновлює ratings.json за допомогою PHP-функцій json_decode і json_encode. Такий підхід

забезпечує простоту реалізації та гнучкість, хоча потребує додаткового захисту для уникнення несанкціонованого доступу до файлу.

Демонстрація сформованих рейтингів у вебсистемі для розв'язування логічних задач відбувається через спеціально розроблений інтерфейс у файлі ratingViewer.html, який взаємодіє з даними, збереженими у ratings.json. Після того, як гравці завершують гру, їхні результати відправляються на сервер через saveRating.php, оновлюючи ratings.json. Цей файл структурований як об'єкт, де ключі відповідають назвам файлів задач ,наприклад, bullsAndCows.html, а значення масивам записів із даними гравців.

Реалізація на стороні клієнта виконується за допомогою JavaScript, який інтегрується з HTML-інтерфейсом ігор (наприклад, Games/*.html). Після завершення гри дані передаються на сервер через асинхронний запит fetch. На рисунку 2.1 зображено блок-схему роботи метода для опису процесу від введення даних до оновлення ratings.json.

Крім основної функціональності збереження результатів, система рейтингу також забезпечує сортування записів за рівнем досягнень, де першочергове значення мають очки. Це дозволяє об'єктивно порівнювати результати гравців і формувати мотиваційний змагальний рейтинг. Такий підхід є особливо ефективним у навчальному середовищі, оскільки заохочує користувачів не лише до правильного, а й до швидкого розв'язання задач. Крім того, система дає змогу зручно аналізувати прогрес користувачів у межах окремих ігор або загальної активності.

Для розширення функціоналу в майбутньому передбачена можливість додавання нових логічних ігор без зміни архітектури системи. Завдяки уніфікованим ключам rate, time, userName та централізованому зберіганню результатів у ratings.json, будь-яка нова гра, що відповідає цим вимогам, автоматично інтегрується в загальну рейтингову систему. Таким чином, структура забезпечує не лише гнучкість і масштабованість, а й мінімізацію витрат на підтримку та розвиток вебсистеми.

Для реалізації рейтингу у вебзастосунку, використовується розроблений метод формування рейтингів і зберігання часу проходження ігор:

1. Користувач завершує виконання логічної задачі.

2. Зчитуються значення rate, time та userName із HTML-елементів гри.

3. Виконується перевірка валідності даних.

4. У разі помилки процес завершується з повідомленням.

5. Дані об’єднуються в об'єкт і серіалізуються у формат JSON.

6. Виконується асинхронний запит fetch() до серверного скрипта saveRating.php.

7. На сервері декодується JSON-об’єкт.

8. Новий запис додається до масиву ratings['games'][$gameFile].

9. Оновлений масив записується у файл ratings.json за допомогою file_put_contents.

10. Перевіряється статус відповіді сервера.

11. У разі невдачі повертається повідомлення про помилку.

12. У разі успіху клієнт отримує підтвердження.

13. Інтерфейс за потреби оновлюється.

14. Процес завершується – рейтинг доступний для перегляду у ratingViewer.html.

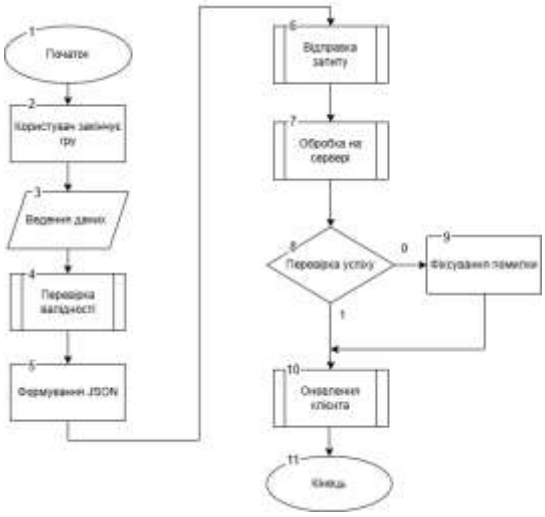


Рисунок 2.1 – Блок-схема алгоритму методу формування рейтингів і зберігання часу проходження ігор

Демонстрація починається з завантаження ratingViewer.html у браузері. Ця сторінка містить інтерфейс із випадаючим списком для вибору гри та таблицею для відображення рейтингів. JavaScript-код на сторінці виконує асинхронний запит через fetch до ratings.json, отримуючи всі доступні дані. Наприклад, код із ratingViewer.html виглядає так: спочатку заповнюється список за допомогою <select id="gameSelect"> назвами ігор із ratings.json, а при виборі гри таблиця #ratingsTable оновлюється відповідними записами.

Інтерфейс стилізовано з використанням Tailwind CSS, де основний контейнер має фон #f0fdf4 (bg-green-50), текст #065f46 (text-green-800), а таблиця включає заголовки «Гравець», «Очки», «Час». При виборі гри ,наприклад, «Бики та Корови». JavaScript сортує записи за rate і відображає їх у таблиці, наприклад: "Player1, 100, 120". Це забезпечує інтуїтивне порівняння результатів.

Демонстрація є динамічною завдяки асинхронним запитам, що дозволяє оновлювати рейтинги без перезавантаження сторінки. Дані з ratings.json читаються через простий HTTP-запит, але для безпеки файл захищений через .htaccess, щоб уникнути прямого доступу. Користувачі бачать актуальні рейтинги в реальному часі, що мотивує до конкуренції та покращення результатів, підтримуючи освітні цілі системи

2.3 Розробка моделі системи

Модель діаграми є інструментом для валідації й верифікації, яка сприяє перевірці всіх необхідних елементів системи представлено на рисунку 2.1. Діаграма класів показує основні об’єкти, які буде використовувати система, їх властивості, поведінку, стереотипи та зв’язки між ними. Допомагає спроєктувати об’єктно-орієнтовану архітектуру, розділити відповідальності між класами, що особливо важливо для масштабованих систем[17].

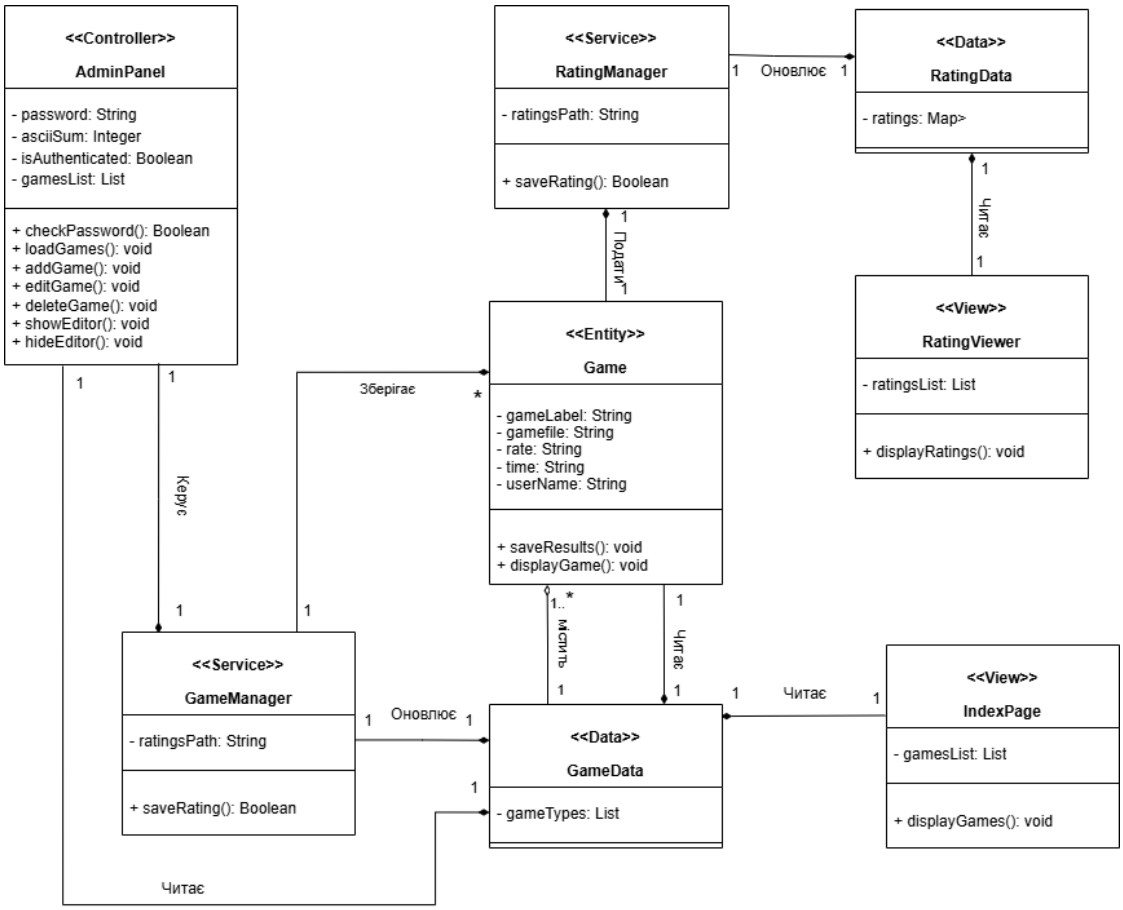


Рисунок 2.2 – Модель роботи системи з використанням діаграми класів

Вебсистема для розв’язування логічних задач, представлена у діаграмі класів, базується на чітко визначених класах, кожен із яких виконує специфічну роль у системі. Клас AdminPanel, позначений стереотипом <<Controller>>, є адміністративною панеллю, що дозволяє керувати іграми, з атрибутами як-от password, asciiSum для автентифікації та gamesList для зберігання списку ігор. Його методи, такі як checkPassword(), перевіряють доступ, тоді як loadGames(), addGame(), editGame(), deleteGame(), showEditor() і hideEditor() забезпечують завантаження, додавання, редагування, видалення ігор та управління інтерфейсом редактора. Клас Game, із стереотипом <<Entity>>, представляє окремі ігри з атрибутами gameLabel, gamefile, rate, time і userName, а методи saveResults() і displayGame() відповідають за збереження результатів і відображення гри.

Інші класи також відіграють ключові ролі в системі. IndexPage, позначений як <<View>>, відображає список ігор через метод displayGames(), використовуючи атрибут gamesList, завантажений із GameData. RatingViewer, із тим самим стереотипом <<View>>, показує рейтинги за допомогою displayRatings(), опираючись на ratingsList із RatingData. На серверному рівні GameManager, із стереотипом <<Service>>, керує іграми через методи saveGame(), deleteGame() і updateParams(), використовуючи paramsPath і gamesFolder, тоді як RatingManager, також <<Service>>, зберігає рейтинги через saveRating() із ratingsPath. Клас GameData, із стереотипом <<Data>>, містить метадані ігор у вигляді gameTypes, а RatingData зберігає рейтинги у вигляді ratings як мапи.

Зв’язки між класами визначають їхню взаємодію в системі. Асоціації, такі як AdminPanel із GameManager через manages, відображають, як адміністративна панель керує серверною логікою, а зв’язок із GameData через reads дозволяє завантажувати дані. IndexPage і Game також читають GameData, тоді як Game подає результати через submits до RatingManager, а RatingViewer читає RatingData. GameManager оновлює GameData і зберігає Game, а RatingManager оновлює RatingData. Композиція між GameData і Game через contains вказує, що GameData є контейнером для ігор, і при його видаленні пов’язані ігри також втрачаються.

Зв’язки, із кардинальністю (наприклад, 1 або 1..*), забезпечують чітке розуміння структури та взаємодії, підкреслюючи архітектуру MVC із клієнтськими (перегляди), серверними (служби) і даними компонентами[18].

Теоретично, UML дозволяє моделювати статичну структуру системи, де класи представляють сутності з атрибутами та поведінкою, а зв’язки – їхні відносини. Асоціації показують прості зв’язки, тоді як композиція вказує на сильну залежність, де одна сутність контролює життєвий цикл іншої. У цій системі стереотипи, як от <<Controller>> чи <<Data>>, додають контекст, а кардинальність уточнює кількість екземплярів, що важливо для реляційних чи JSON-систем. Практично, ці класи та зв’язки реалізують функціонал від

керування іграми до відображення рейтингів, забезпечуючи масштабованість і зрозумілість.

Система демонструє інтеграцію клієнтських і серверних компонентів, де AdminPanel, Game, IndexPage і RatingViewer забезпечують інтерфейс, а GameManager і RatingManager обробляють логіку, опираючись на дані з GameData і RatingData. Композиція між GameData і Game підкреслює централізоване зберігання метаданих, тоді як асоціації відображають динамічну взаємодію через HTTP-запити чи читання/оновлення. Ця структура, узгоджена з UML, підтримує освітні цілі системи, мотивуючи гравців через рейтинги та спрощуючи адміністрування.

2.4 Розробка алгоритму роботи системи та діаграм діяльності з процесною схемою

Для розробки програмних модулів вебсистеми для розв'язування логічних задач необхідно розробити загальний алгоритм, згідно якого буде працювати вебсистема (рис. 2.3). На початку користувач вводить ім’я. Якщо обрано адміністративну панель, система запитує пароль та перевіряє його; у разі помилки – виводиться повідомлення, у разі успіху – адміністратор отримує доступ до функцій редагування або додавання нових задач.

У користувацькому сценарії після введення імені користувач обирає тип головоломки, система генерує задачу, і користувач розв’язує її. Далі здійснюється перевірка відповіді: при правильному рішенні результат зберігається, оновлюється рейтинг і відображаються підсумки. У разі програшу виводиться повідомлення та правильна відповідь. В обох випадках користувач вирішує, чи завершити сесію.

Адміністратор після успішної валідації може створити нову задачу, яка інтегрується в загальну систему. Після цього він вирішує, чи залишитися в панелі, чи завершити роботу. Схема охоплює повний цикл взаємодії як для користувача, так і для адміністратора.

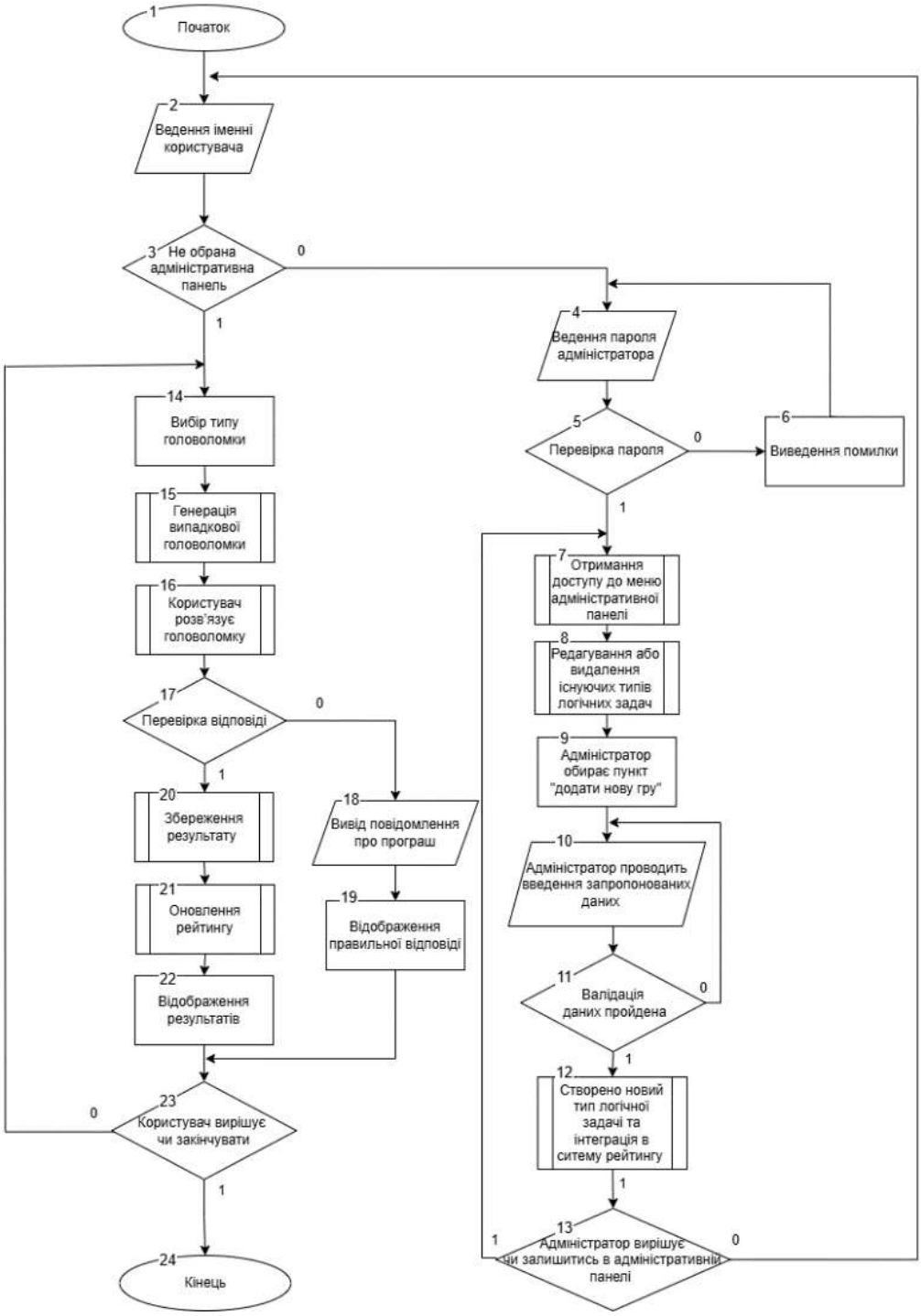


Рисунок 2.3 – Загальний алгоритм роботи вебсистеми

Модуль адміністрування: Відповідає за керування іграми (створення, редагування, видалення). Включає adminPanel.html (інтерфейс і JavaScript-код), saveGame.php (серверна обробка) і частину params.json (метадані ігор). Цей модуль забезпечує автентифікацію адміністратора та оновлення списку ігор.

Діаграма діяльності – це тип UML-діаграми, який моделює послідовність дій, логіку управління потоком і гілкування процесів у системі. Вона показує, як користувач або система переходить від однієї дії до іншої, включаючи альтернативні та паралельні шляхи виконання. Такі діаграми часто використовують для візуалізації бізнес-процесів, алгоритмів, сценаріїв використання або логіки роботи модулів. Основними елементами є дії (прямокутники з заокругленими кутами), умови (ромби), початкові та кінцеві точки, а також стрілки, що позначають напрямок потоку[19].

Діаграма діяльності модуля адміністрування(рис. 2.4) описує послідовність дій, які виконує адміністратор під час роботи з інтерфейсом керування іграми або задачами. Початковою дією є відображення форми автентифікації. Адміністратор вводить пароль, після чого система перевіряє його валідність. У разі неправильного пароля процес завершується. Якщо ж пароль правильний, відбувається завантаження списку ігор із файлу params.json за допомогою функції loadGames().

Після успішної автентифікації адміністратор отримує доступ до основного інтерфейсу системи, де може виконувати три основні дії: перегляд і видалення задачі, редагування вже наявного запису або додавання нової гри/задачі. У кожному з цих випадків передбачено підтвердження дії з боку адміністратора. Для додавання нової гри чи задачі також обов’язкова перевірка коректності введених даних.

Після підтвердження будь-якої дії система відправляє дані на сервер за допомогою запиту fetch до скрипта saveGame.php, який оновлює файл params.json відповідно до внесених змін. На наступному етапі перевіряється статус відповіді сервера. Якщо відповідь має статус 200 OK, відбувається оновлення інтерфейсу, що свідчить про успішне збереження. Якщо ж статус відрізняється від 200, адміністратору виводиться повідомлення про помилку збереження. Після завершення усіх дій процес адміністрування завершується.

Ця діаграма діяльності наочно демонструє логіку роботи адміністративного модуля, включаючи всі варіанти дій, перевірки і можливі відхилення від основного сценарію.

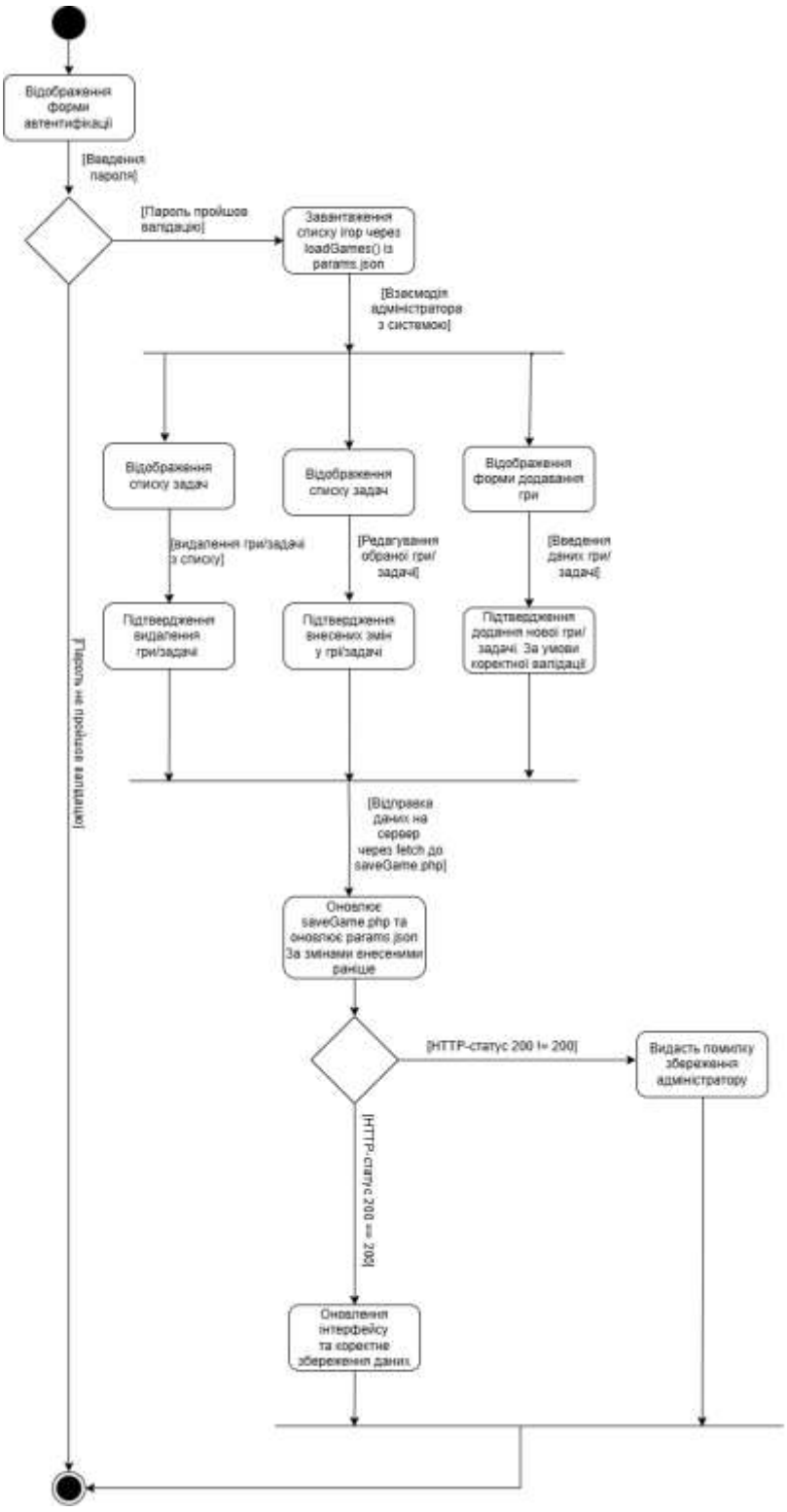


Рисунок 2.4 – Діаграма діяльності модуля адміністрування

Модуль ігор: Зосереджений на ігровому процесі та збереженні результатів. Включає файли Games/*.html (наприклад, bullsAndCows.html), що містять HTML-структуру, JavaScript-код для відображення гри та відправки даних (fetch до saveRating.php), а також стилі з styles.css.

Діаграму діяльності модуля ігор/ задач наведено на рисунку 2.5.

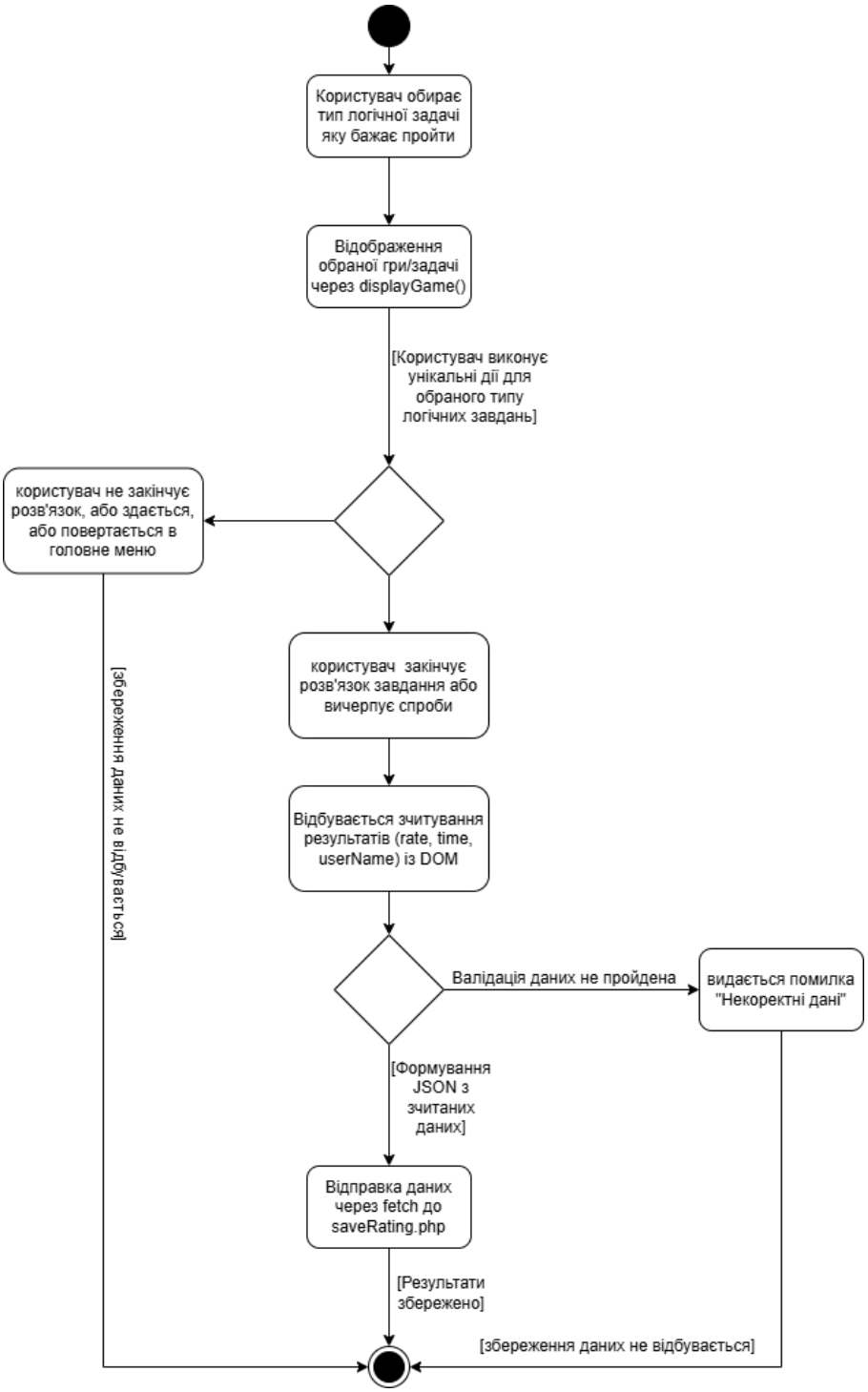


Рисунок 2.5 – Діаграма діяльності модуля ігор

Модуль рейтингів: Відповідає за обробку та демонстрацію рейтингів. Включає ratingViewer.html (інтерфейс для відображення таблиці рейтингів), saveRating.php (серверна логіка збереження) і ratings.json (сховище рейтингів).

Діаграму діяльності модуля рейтингів наведено на рисунку 2.6.

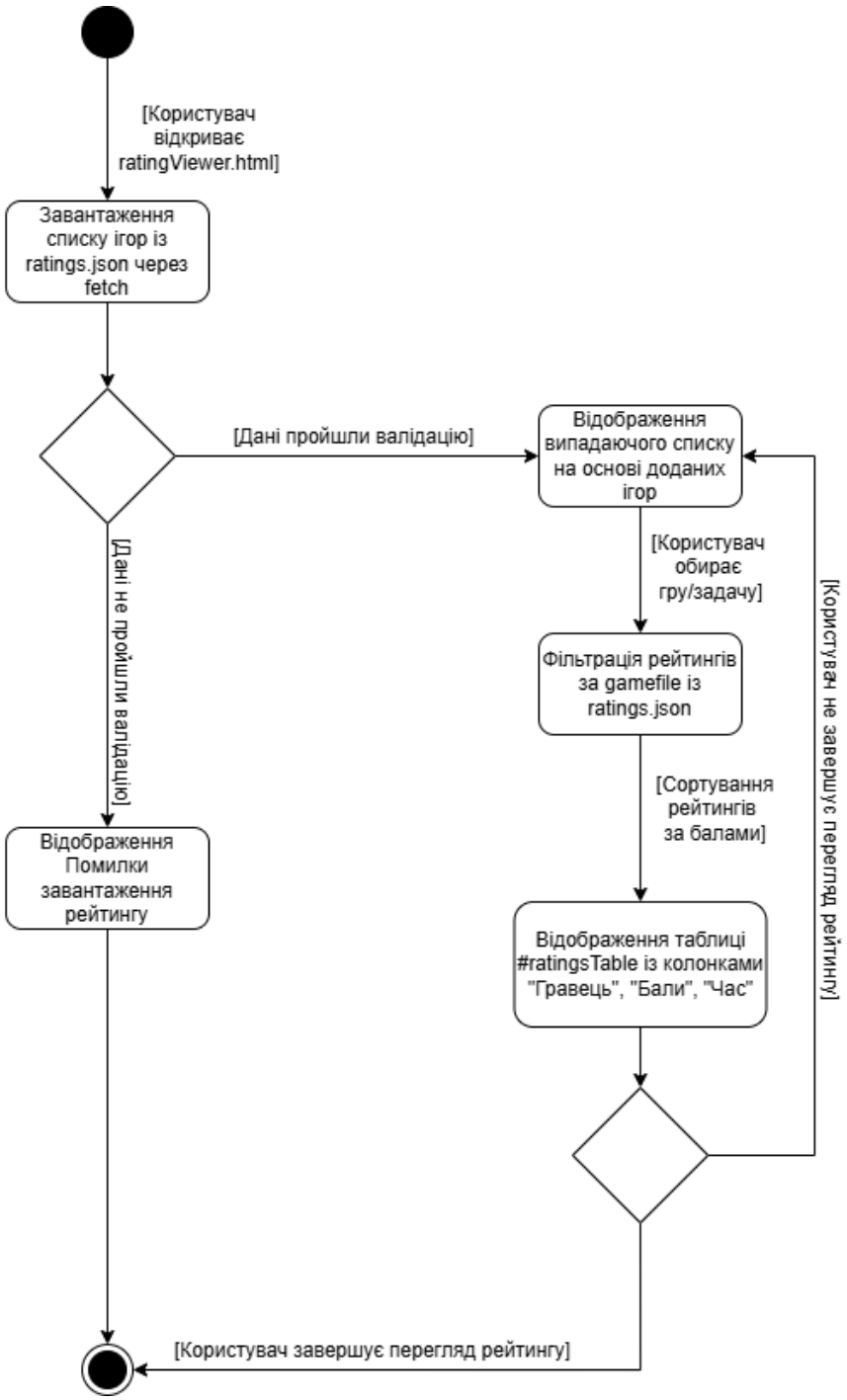


Рисунок 2.6 – Діаграма діяльності модуля рейтингів

Модуль даних. Централізоване сховище метаданих і рейтингів. Включає params.json (список ігор) і ratings.json (результати гравців). Цей модуль є пасивним, але використовується іншими для читання та запису.

Процесна схема[20] – це послідовне зображення дій, що ведуть до результату. Вона особливо корисна для візуального розуміння логіки роботи програмного модуля, бізнес-процесу або навіть просто алгоритму.

Процесна схема модуля даних наведено на рисунку 2.7. Ця діаграма описує оновлення ratings.json аналогічно для params.json.

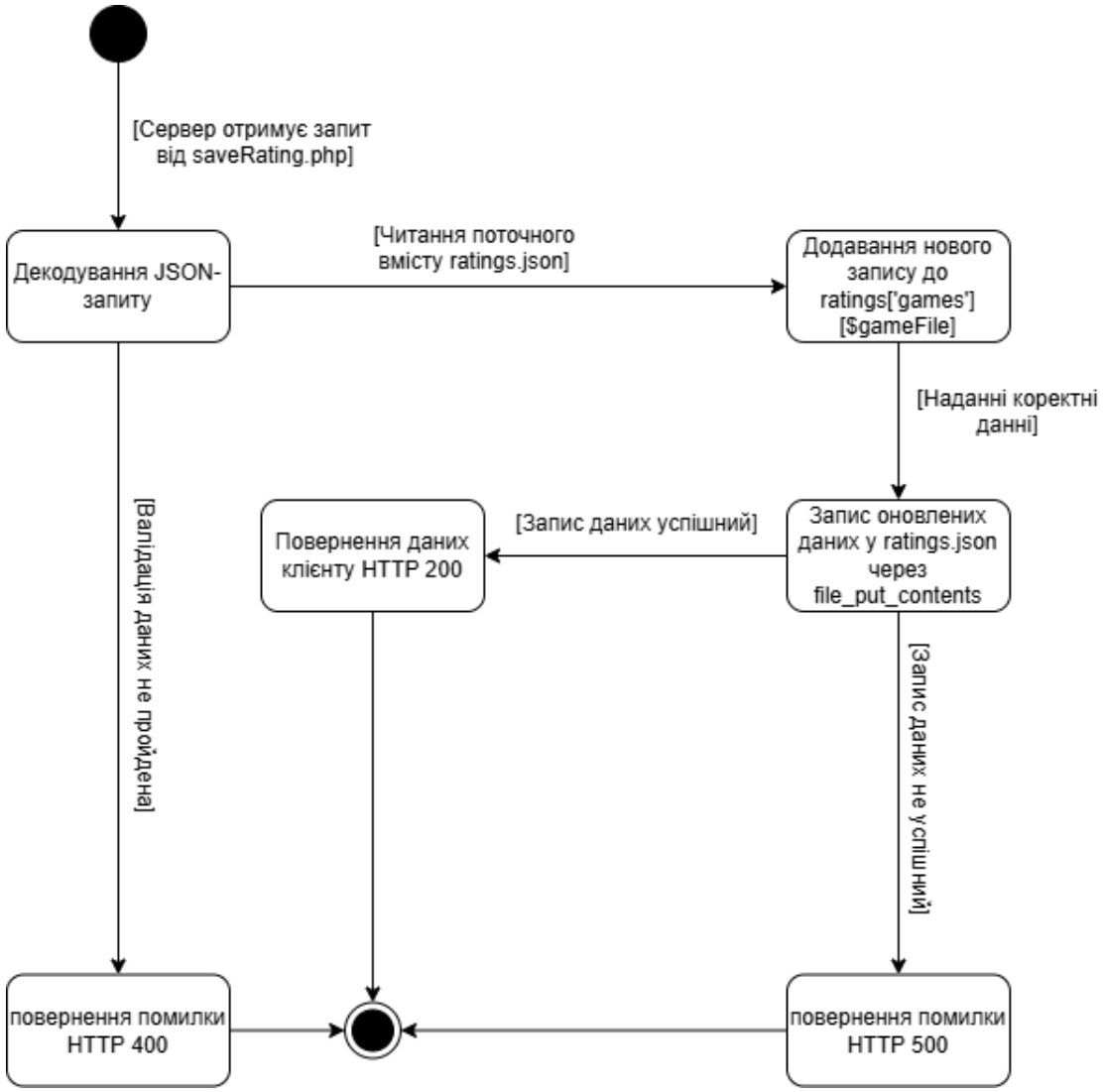


Рисунок 2.7 – Процесна схема модуля даних

2.5 Висновки

У другому розділі бакалаврської кваліфікаційної роботи здійснено комплексну розробку концептуальної основи вебсистеми для розв’язування логічних задач. Проведено детальний аналіз вхідних даних системи, визначено ключові параметри взаємодії користувача з інтерфейсом, зокрема ім’я гравця, обраний тип задачі, параметри складності та результати проходження. Визначено функціональні ролі кожного модуля системи – ігрового, адміністративного, рейтингового та модуля даних та описано їх взаємодію через JSON і PHP.

Запропоновано метод формування рейтингових записів, що базується на кількісній оцінці ефективності дій користувача (бали та час виконання) та реалізується через серверну логіку із збереженням у централізованому файлі ratings.json. Розроблено модель системи у вигляді UML-діаграми класів, яка структуровано відображає основні компоненти системи, їхні атрибути, методи та зв’язки.

Крім того, створено алгоритм роботи вебсистеми та побудовано діаграми діяльності й процесну схему для кожного ключового модуля. Це дозволило формалізувати логіку функціонування системи, описати ключові сценарії взаємодії, зокрема з боку користувача й адміністратора. Усі розроблені рішення спрямовані на забезпечення зручного, ефективного та масштабованого функціоналу для користувачів, а також простоти адміністрування й гнучкості системи для подальшого розвитку.

3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ ВЕБСИСТЕМИ ДЛЯ РОЗВ'ЯЗАННЯ ЛОГІЧНИХ ЗАДАЧ

3.1 Варіантний аналіз і обґрунтування вибору засобів реалізації програми

Для реалізації вебсистеми з логічними іграми було проведено аналіз можливих варіантів засобів розробки, виходячи з основних вимог: підтримка кількох типів головоломок, збереження результатів, формування рейтингу користувачів, простота розгортання та орієнтація на навчальні цілі без потреби в складній серверній інфраструктурі.

На стороні клієнта (інтерфейс користувача) було обрано комбінацію HTML, CSS та чистого JavaScript, оскільки вона забезпечує достатній функціонал для побудови динамічного інтерфейсу та не потребує додаткових інструментів для збірки, таких як Webpack або Vite. Крім того, цей підхід простий у засвоєнні, що робить його ідеальним для студентських проєктів. Було розглянуто також використання фреймворків, таких як React чи Vue, однак вони вимагають глибших знань і складнішої конфігурації, що є надлишковим для цілей даного проєкту.

У ролі серверної частини системи було обрано PHP, що є класичним вибором для легких вебсайтів і добре підтримується навіть на найпростіших хостингах. На відміну від Node.js або Python (Flask/Django), PHP не потребує попередньої інсталяції додаткових серверів або віртуальних середовищ. Для збереження даних про гру (результатів, балів, імен користувачів, дати) було вирішено використовувати JSON-файл, оскільки він є простим у використанні, не потребує налаштування бази даних і забезпечує достатній рівень наочності та гнучкості для малих проєктів. JSON-файл зручно обробляється в PHP (через json_encode і json_decode), і його легко оновлювати без складної транзакційної системи [21].

Щодо формату збереження, альтернативами розглядалися MySQL [22] (для структурованих даних) та LocalStorage (для клієнтського зберігання).

Однак перший варіант є надлишковим, а другий – не дозволяє централізовано збирати результати для формування рейтингу між усіма користувачами. Таким чином, JSON-файл на сервері є оптимальним вибором.

Для виведення рейтингу та історії проходження було обрано просту реалізацію через DOM-елементи – таблиці або списки, сформовані динамічно через JavaScript [23]. Це рішення не потребує сторонніх бібліотек (як-от Chart.js чи D3.js), які хоч і дозволяють створити візуально привабливі графіки, проте ускладнюють структуру проєкту. В межах завдання достатньо простого табличного виводу з підтримкою сортування та групування результатів.

Отже, поєднання мов програмування JavaScript та PHP є доцільним для реалізації вебсистеми логічних задач, оскільки забезпечує ефективну взаємодію клієнтської та серверної частин: JavaScript відповідає за динамічність інтерфейсу, а PHP – за обробку даних і збереження результатів. Такий підхід дозволяє системі приймати структуровані вхідні дані (ім’я користувача, тип гри, спроби тощо), обробляти їх, формувати об’єкт із результатом гри й зберігати у форматі JSON, що в подальшому використовується для генерації історії проходження та рейтингу. Таким чином, система стає не лише функціональною, а й зручною для користувачів, відповідаючи сучасним вимогам веброзробки.

Загалом, обрані інструменти (HTML/CSS/JS + PHP + JSON) дозволяють швидко й ефективно створити вебсистему з підтримкою інтерактивної логіки, історії проходження та рейтингів користувачів, з можливістю локального або серверного розгортання без потреби в складних зовнішніх сервісах. Такий підхід забезпечує баланс між функціональністю, простотою реалізації та відповідністю навчальним цілям.

Вибір Visual Studio Code для розробки вебсистеми розв’язування логічних задач обґрунтований рядом технічних, практичних і організаційних переваг, які відповідають потребам проєкту, включаючи створення модулів ігор, адміністрування, рейтингів і даних. Розробка охоплює HTML, JavaScript, PHP, JSON і CSS, що потребує універсального і легкого інструменту.

Visual Studio Code (VS Code) – це безкоштовне, легке та потужне інтегроване середовище розробки (IDE), розроблене компанією Microsoft [24]. VS Code забезпечує високу продуктивність і легкість, що критично для роботи з невеликими проєктами, такими як дана система. Завдяки кросплатформності (підтримка Windows, macOS, Linux) і мінімальним системним вимогам (близько 200 МБ оперативної пам'яті) інструмент ефективно працює навіть на слабких машинах, що полегшує розробку у навчальних закладах, де можуть використовуватися різні пристрої. Швидкий запуск і низьке споживання ресурсів дозволяють зосередитися на кодуванні без затримок.

Безкоштовність і відкритий код VS Code зробили інструмент доступним для розробки, усуваючи фінансові бар'єри. Велика спільнота розробників забезпечує регулярні оновлення і доступ до тисяч розширень через Marketplace, що дозволяє швидко вирішувати проблеми, наприклад, через пошук рішень у документації або форумах.

Альтернативи, такі як PhpStorm, відхилені через високу вартість близько $99/рік і більшу ресурсоємність яка складає понад 1 ГБ оперативної пам'яті, що ускладнює використання на слабких машинах. Sublime Text, хоча й швидкий, не має такої глибокої інтеграції з Git і розширень, як VS Code.

Отже вибір VS Code обґрунтований балансом між функціональністю, продуктивністю і доступністю, що ідеально відповідає потребам розробки даної системи

3.2 Розробка програмного модуля ігор

Модуль ігор представляю собою всі типи задач розробленні для вебсистеми. На момент розробки модуля повноціно реалізовано два типи логічних задач «Бики та корови» та «Задача Енштейна». Модуль ігор не постійний тому що за допомогою модуля адміністрування реалізовано редагування, видалення та додавання нових типів логічних задач.

Модуль ігор вебсистеми для розв'язування логічних задач, зокрема задача «Бики та корови», реалізовано через файл bullsAndCows.html, який

поєднує HTML-структуру, JavaScript-логіку та стилі Tailwind CSS, а також використовує saveRating.php для серверної обробки та ratings.json для збереження результатів; додатково читаються метадані з params.json. У bullsAndCows.html функція generateNumber() створює унікальне чотиризначне число, startTimer() відстежує час гри, checkGuess() перевіряє введення, підраховує "биків" і "корів", обчислює rate (100 - 5 * спроб, мінімум 10), а saveResults() відправляє JSON із gamefile, userName, rate і time через fetch до saveRating.php. Далі наданий більш детальний опис функцій які є спільними для всіх задач в модулі, надано приклад розробки задачі для вебсистеми.

Аналізуючи структуру та логіку для розробки «Бики та корови» та «Задача Ейнштейна», виділено спільні функції, що забезпечують уніфікований підхід до реалізації ігор, їхньої інтерактивності та інтеграції з іншими модулями. Ці функції відображають спільну архітектуру, адаптовану цілей системи.

Першою спільною функцією (рис. 3.1) є ініціалізація гри, яка запускає основний інтерфейс гри після її завантаження. У bullsAndCows.html це відбувається через автоматичне відображення поля введення числа та підказок при виклику HTML-контенту, тоді як для «Задачі Ейнштейна» передбачається аналогічна ініціалізація, наприклад, відображення поля для введення відповідей і підказок. Ця функція налаштовує UI, зчитуючи метадані з params.json (наприклад, gameLabel і gamefile), і забезпечує базову взаємодію з гравцем.

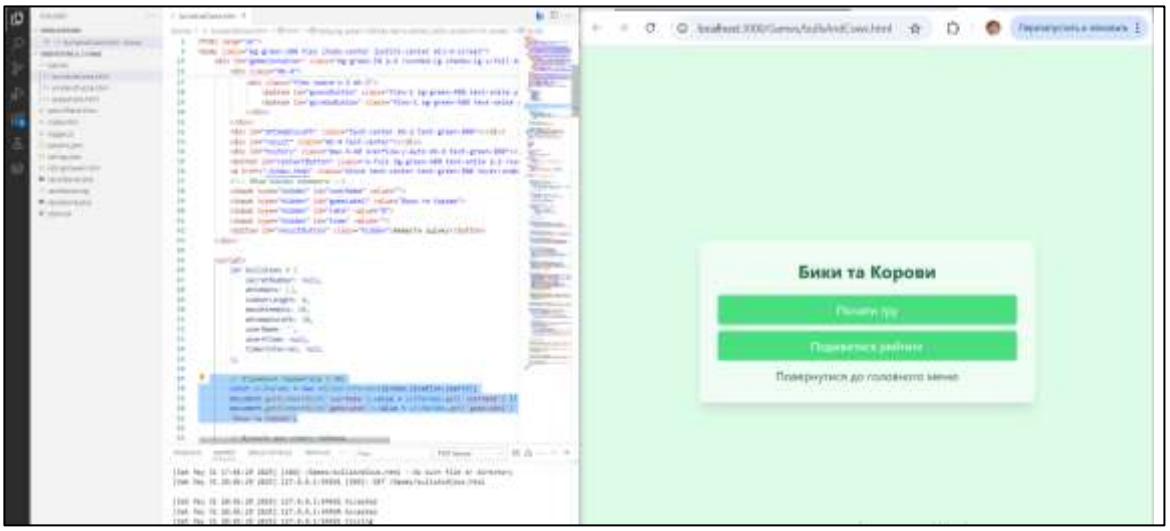


Рисунок 3.1 – Ініціалізація сторінки гри

Другим спільною функцією (рис. 3.2) є обробка дій гравця checkInput() або еквівалент, який аналізує введення користувача та повертає зворотний зв’язок. У «Биках та коровах» це реалізовано через checkGuess(), що перевіряє введене число, підраховує «биків» і «корів» та оновлює підказки. Для «Задачі Ейнштейна» передбачається аналогічний метод, який перевіряє логічні відповіді і видає підказки про правильність. Обидва методи включають валідацію введення (наприклад, унікальність цифр у «Биках та коровах» чи коректність формату відповіді в «Задачі Ейнштейна») і базуються на асинхронній логіці для динамічного оновлення.

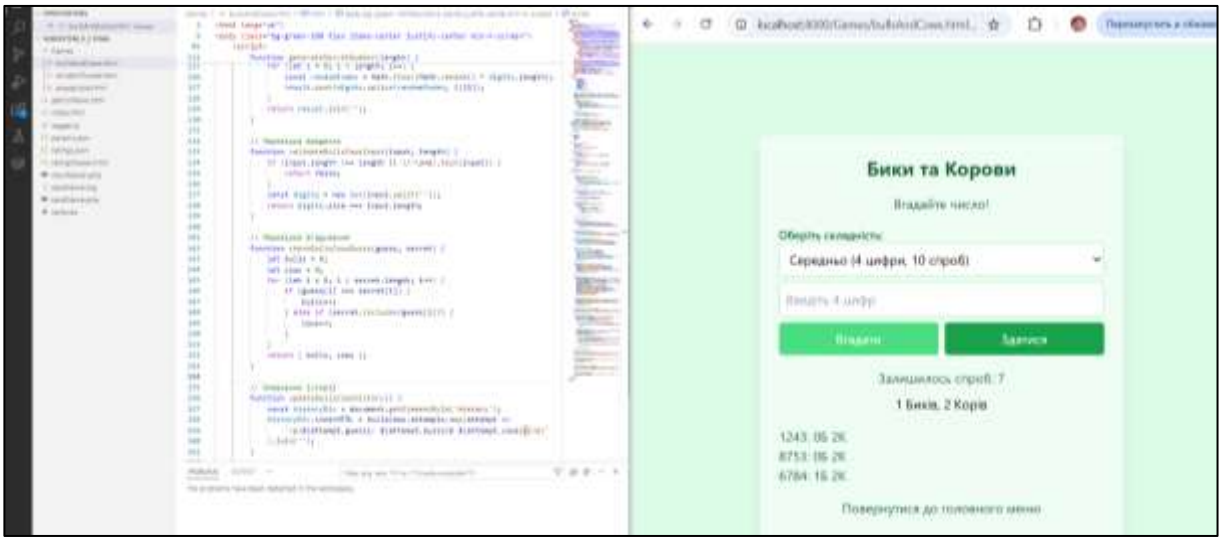


Рисунок 3.2 – Обробка дій гравця

Третьою спільною функцією (рис. 3.3) є обчислення та збереження результатів calculateResults() або еквівалент, який визначає rate (бали) і time (час проходження) після завершення гри. У bullsAndCows.html це реалізовано через обчислення rate (100 - 5 * спроб, мінімум 10) і зчитування time з таймера після вгадування числа; для «Задачі Ейнштейна» передбачається аналогічний підхід, де rate може залежати від кількості правильних відповідей, а time – від тривалості вирішення. Цей метод активується при завершенні гри і підготовляє дані для збереження.

Четвертою спільною функцією (рис. 3.3) є збереження результатів (saveResults()), який інтегрує гру з модулем рейтингів. У bullsAndCows.html функція saveResults() формує JSON-об’єкт із gamefile, userName, rate і time, відправляє його через fetch до saveRating.php, де дані додаються до ratings.json. Для "Задачі Ейнштейна" передбачається ідентична логіка, адаптована до її формату, із валідацією userName і обробкою помилок, забезпечуючи уніфіковане збереження результатів у ratings.json.

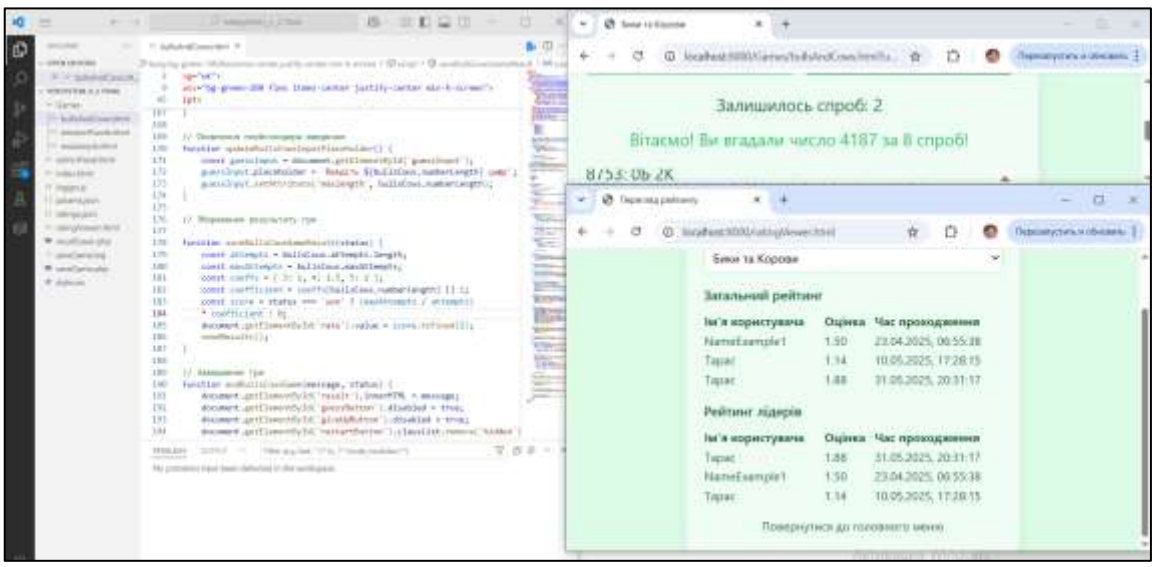


Рисунок 3.3 – Обчислення та збереження результатів

П’ятою спільною(рис.3.4) функцією є обробка помилок і валідація validateInput() або еквівалент, який гарантує коректність даних перед їхньою обробкою. У «Биках та коровах» це реалізовано через validateGuess()

(перевірка унікальності цифр) і validateUserData(); для «Задачі Ейнштейна» передбачається аналогічний метод, який перевіряє формат відповідей і валідність імені гравця, із виведенням повідомлень про помилки через DOM-елементи (наприклад, #errorMessage).

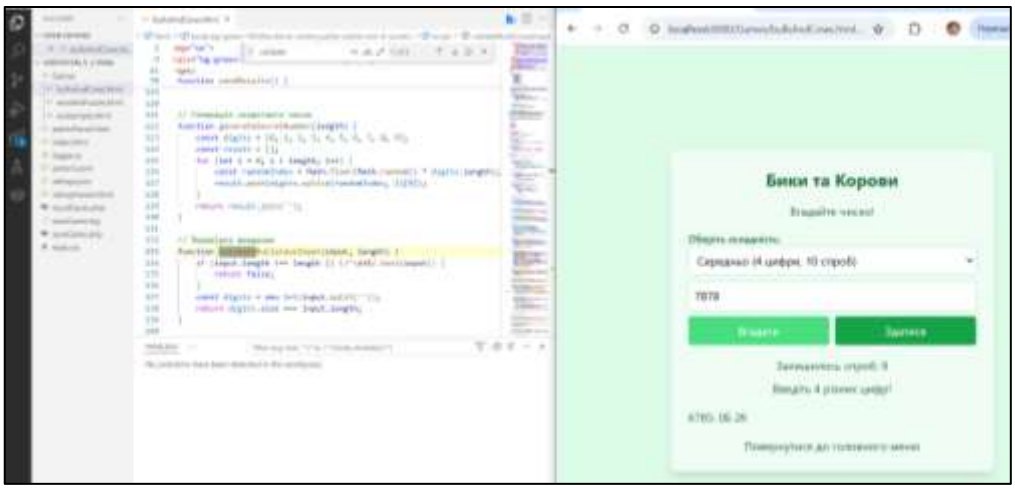


Рисунок 3.4 – Обробка помилок та валідація

Ці спільні функції забезпечують послідовність у розробці ігор, дозволяючи легко додавати нові логічні задачі, такі як «Задача Ейнштейна», до системи. Вони інтегруються з модулем даних (params.json, ratings.json) через асинхронні запити, підтримують єдиний стиль Tailwind CSS і сприяють освітнім цілям, мотивуючи гравців до конкуренції через рейтинги.

Для кожної задачі також існують унікальні функції які реалізують основний функціонал задач. Для гри «Бики та корови» унікальною функцією є countBullsAndCows(), яка обчислює кількість «биків» правильні цифри на правильних позиціях і «корів» правильні цифри на неправильних позиціях під час вгадування чотиризначного числа. Ця функція аналізує введене гравцем число, порівнюючи його з цільовим числом, згенерованим через generateNumber(), і повертає результат у форматі { bulls: number, cows: number}. Функція є ключова для розв’язання, оскільки забезпечує зворотний зв’язок, необхідний для логічного аналізу, і не має аналогів у інших задач через специфічність правил «Биків та корів».

На рисунку 3.5 зображено алгоритм генерування випадкового числа для задачі «Бики та Корови» полягає у формуванні секретного числа заданої довжини (зазвичай від 3 до 5 цифр), яке не містить повторень. Для цього спочатку створюється список цифр від 0 до 9, після чого у циклі length і разів випадково вибирається одна цифра з цього списку, додається до результату та видаляється з початкового набору, щоб уникнути повторів. У підсумку отримується рядок із унікальних цифр – секретне число, яке користувач має вгадати.

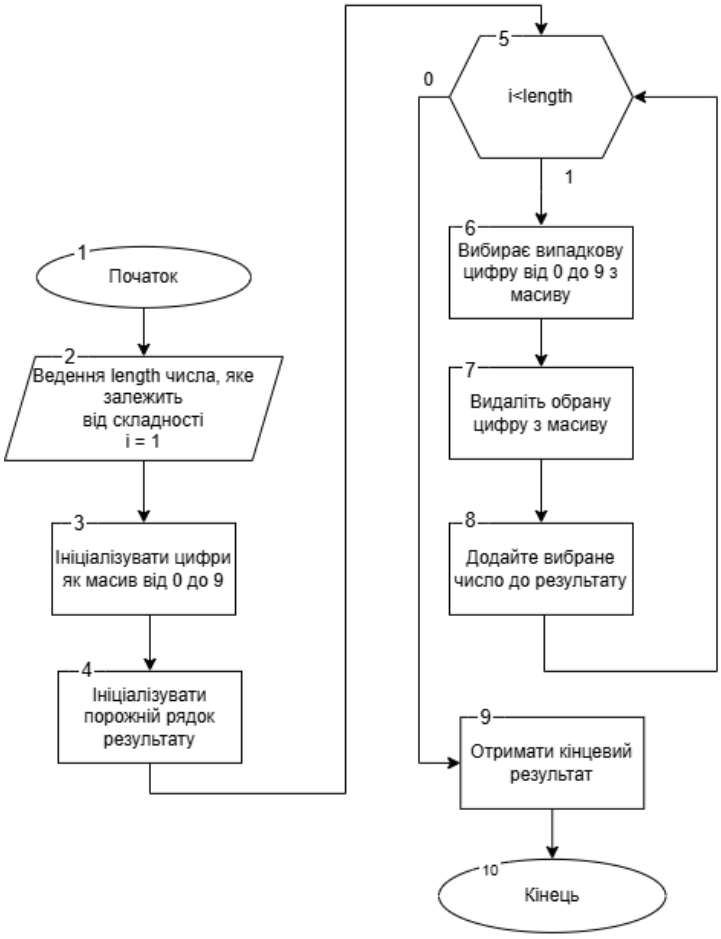


Рисунок 3.5 – Алгоритм генерування випадкового числа для вирішення задачі «Бики та Корови»

Для «Задача Ейнштейна» унікальною функцією є evaluateLogicPuzzle(), яка перевіряє логічні відповіді гравця на основі заданих умов наприклад, порядок будинків, національність, колір, напій тощо і визначає правильність

розв’язання. Ця функція порівнює введені дані з заздалегідь визначеним рішенням (наприклад, масивом правильних відповідей), враховуючи логічні обмеження задачі, і повертає рівень правильності (наприклад, відсоток або кількість правильних пар), що впливає на обчислення rate. Її унікальність полягає в обробці складних умов і логічних зв’язків, відмінних від числового аналізу в «Биках та коровах».

Для «Задачі Ейнштейна» унікальною функцією є generateEinsteinPuzzle(), яка створює або завантажує набір умов задачі, що зображено на рисунку 3.6. Ця функція забезпечує динамічність задачі, дозволяючи варіювати складність, і є специфічною для логічних головоломок, на відміну від числової гри «Бики та корови», де умови статичні.

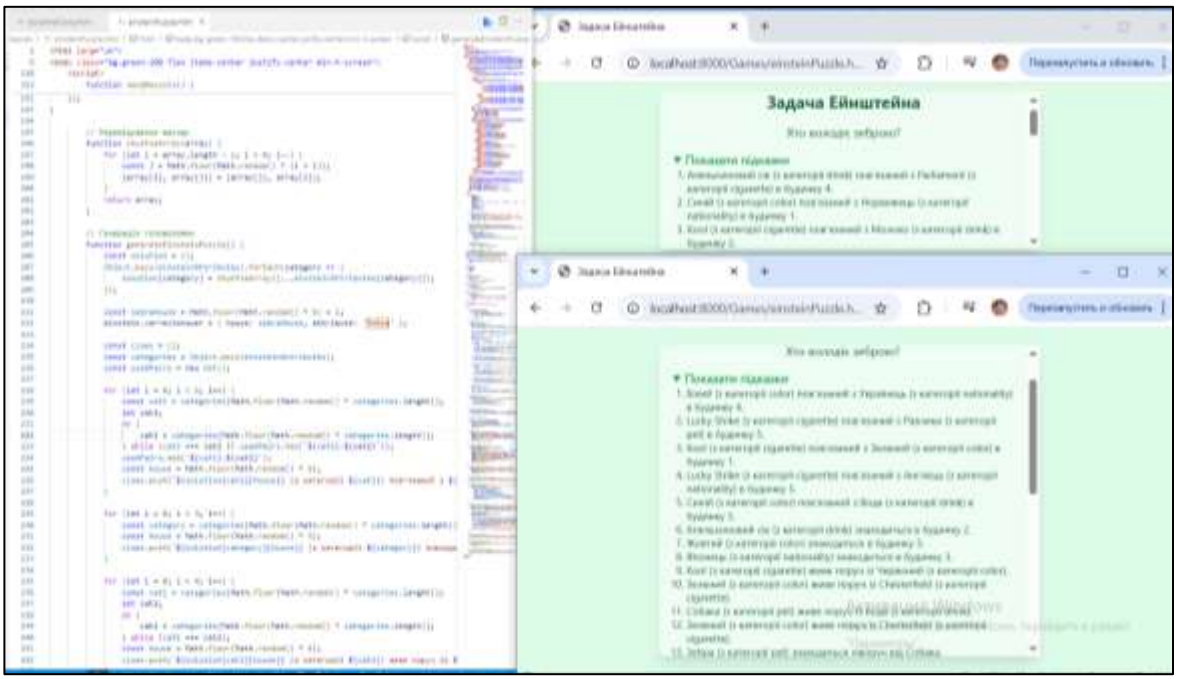


Рисунок 3.6 – Генерування випадкової «Задачі Енштейна» з різними умовами вирішення

Остання унікальна функція для «Задачі Ейнштейна» є provideHint(), яка генерує підказки на основі часткової правильності відповідей гравця (наприклад, вказує на одну правильну пару будинків), допомагаючи у вирішенні складної логічної задачі. У «Биках та коровах» підказки надаються

автоматично через «биків» і «корів», тому додаткова функція підказок не потрібна, роблячи provideHint() унікальною для «Задачі Ейнштейна».

Ці унікальні функції відображають специфіку кожної гри: «Бики та корови» зосереджені на числовому аналізі та зворотному зв’язку, тоді як «Задача Ейнштейна» грунтується на логічних висновках і підказках. Інтегруючись з спільними функціями, зберігаючи уніфіковану архітектуру, і дозволяють адаптувати систему під нові ігри/задачі, зберігаючи їхню індивідуальність та функціональність. Отже модуль ігор складається з розроблених логічних типів задач, які складаються з спільних та унікальних функцій необхідні для роботи завдання.

3.3 Розробка програмного модуля адміністрування

Розробка модуля адміністрування вебсистеми для розв’язування логічних задач забезпечила повноцінне керування іграми через файл adminPanel.html, який інтегрує HTML-структуру, JavaScript-логіку та стилі Tailwind CSS, із серверною логікою в saveGame.php і сховищем даних у params.json. Цей модуль надає можливості автентифікації адміністратора, перегляду списку ігор, додавання нових, редагування існуючих і видалення їх, із валідацією даних і захистом від помилок, що забезпечує зручність адміністрування.

Функції клієнтської частини модуля реалізовано в adminPanel.html для забезпечення інтерактивності та керування інтерфейсом.

Функція authenticateAdmin() виконує автентифікацію адміністратора шляхом отримання пароля з поля(рис. 3.7) #passwordInput, обчислення суми ASCII символів, при успішній перевірці приховує форму #authForm, відображає редактор #editor і викликає loadGames().

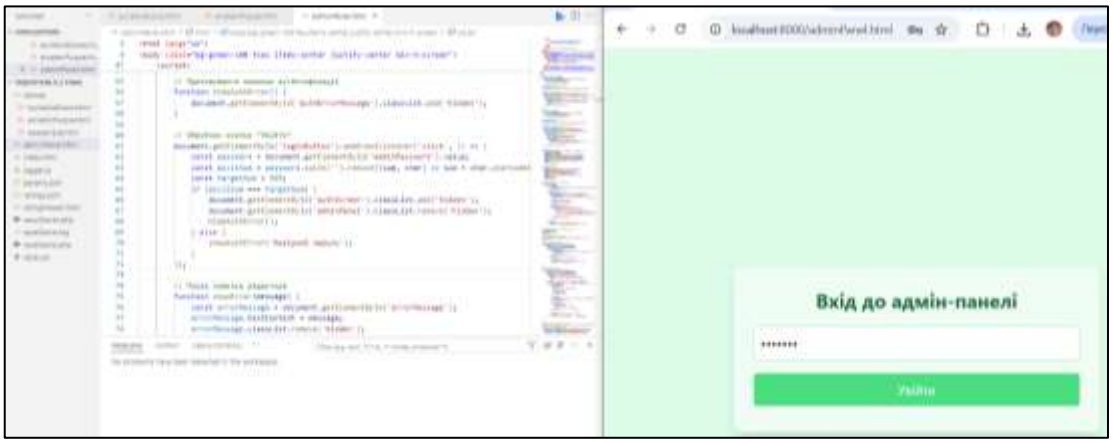


Рисунок 3.7 – Автентифікація адміністратора шляхом отримання пароля з поля

Функція loadGames() завантажує список ігор(рис. 3.8) із params.json через асинхронний fetch-запит, парсить JSON, сортує ігри за gameLabel у алфавітному порядку, генерує HTML для #gamesList із кнопками "Редагувати" і "Видалити", відображає «Немає ігор» у разі порожнього списку, логуючи помилки через console.error. Функція addGame() додає нову гру, валідуючи gameLabel від 1 до 50 символів і gamefile має закінчуватися на «.html», та мати лише латинські символи. Потім перевіряє дублювання gamefile у params.json, відправляє JSON через fetch до saveGame.php і при успіху оновлює список через loadGames(), інакше видає попередження про помилки через alert.

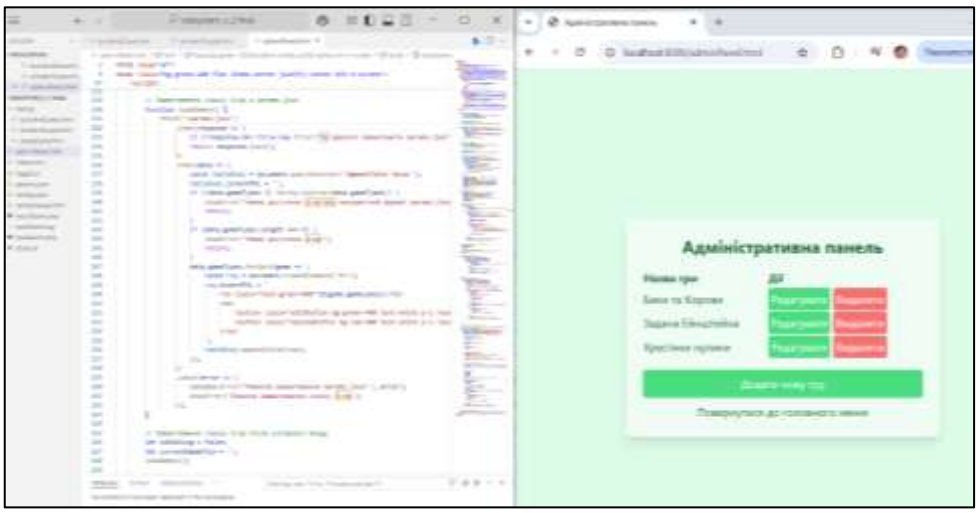


Рисунок 3.8 – Завантаження списку логічних ігор в адміністративній панелі

Функція editGame(gamefile) забезпечує редагування існуючої гри шляхом пошуку за gamefile у списку, заповнення форми #addGameForm даними (gameLabel, gamefile), блокування поля gamefile для уникнення конфліктів, зміни тексту кнопки на «Зберегти зміни», відправлення оновлених даних до saveGame.php із параметром action: 'update', із повідомленням «Гра не знайдена» у разі помилок.

Функція deleteGame(gamefile) видаляє гру, відправляючи JSON-запит {action: 'delete', gamefile } через fetch до saveGame.php після підтвердження через confirm(), і при успіху оновлює список через loadGames(), інакше відображає «Помилка видалення».

Функції showEditor() і hideEditor() керують видимістю форми #addGameForm, очищаючи поля при показі і змінюючи текст кнопки залежно від режиму додавання чи редагування, що сприяє зручності роботи.

Серверна частина в «saveGame.php» обробляє запити на додавання, редагування і видалення ігор, декодуючи JSON-запит, валідуючи дані (наявність полів, формат gamefile), оновлюючи params.json (додавання, оновлення або видалення запису), повертаючи JSON-відповідь ({ status: 'success' } або { status: 'error', message: '...' }), із перевіркою розміру файлу і логуванням операцій для відстеження змін.

3.4 Розробка програмного модуля рейтингів

Розробка програмного модуля рейтингів вебсистеми для розв’язування логічних задач забезпечила відображення та збереження результатів ігор через файл ratingViewer.html, який інтегрує HTML-структуру, JavaScript-логіку та стилі Tailwind CSS, із серверною логікою в saveRating.php і сховищем даних у ratings.json. Цей модуль дозволяє завантажувати список ігор, відображати рейтинги для обраної гри, сортувати їх за очками та зберігати нові результати, із валідацією даних і обробкою помилок, що забезпечує зручність для користувачів.

Функції клієнтської частини модуля реалізовано в ratingViewer.html для забезпечення інтерактивності та відображення даних. Функція loadGames() завантажує список ігор із ratings.json через асинхронний fetch запит, парсить JSON, витягує ключі об’єкта ratings.games, генерує <option> для елемента <select id="gameSelect">, сортуючи ігри за алфавітом, і відображає «Немає ігор» у #gameSelectMessage, якщо список порожній, логуючи помилки через console.error.

На рисунку 3.9 зображено функцію displayRatings() відображає рейтинги для обраної гри, отримуючи gamefile із #gameSelect, фільтруючи записи за gamefile, сортуючи їх за rate у спадному порядку, генеруючи рядки для таблиці #ratingsTable із колонками «Гравець», «Очки», «Час», і відображаючи «Немає даних» у #ratingsTableMessage, якщо записи відсутні, із динамічним оновленням при зміні вибору гри.

Функція clearTable() очищає таблицю #ratingsTable перед оновленням(рис 3.9), видаляючи всі рядки, крім заголовка, що забезпечує коректне відображення нових даних без дублювання, і застосовується перед кожним викликом displayRatings().

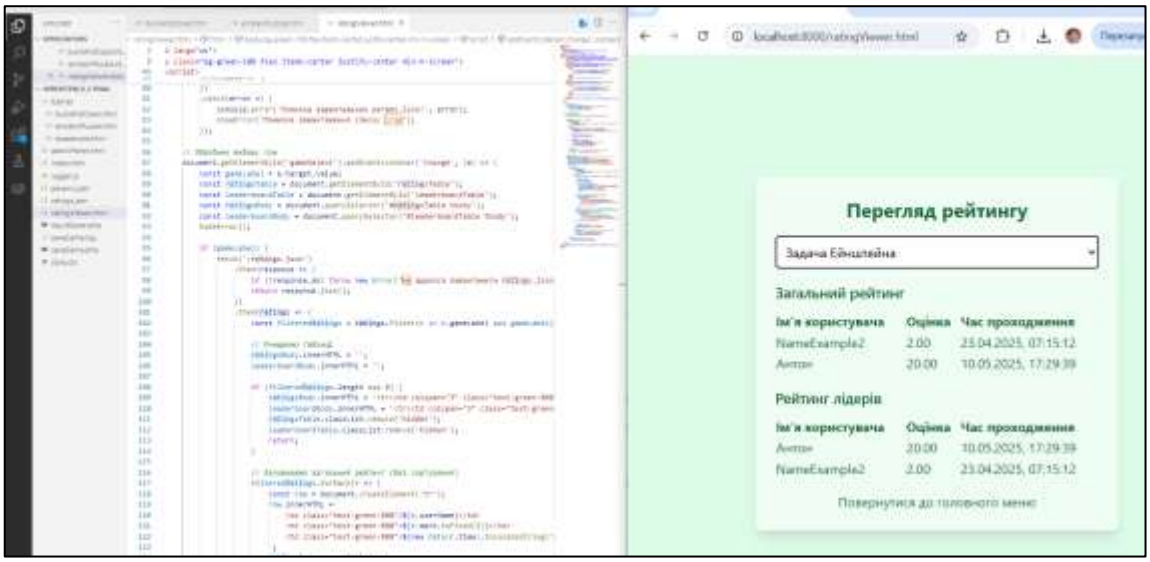


Рисунок 3.9 – Відображення роботи функції displayRatings() та clearTable()

Функція handleError(error) обробляє помилки асинхронних запитів, логуючи їх через console.error і відображаючи повідомлення «Помилка завантаження даних» у #errorMessage, що підвищує інформованість користувача.

Функція sortRatings(ratings) сортує масив рейтингів за полем rate у спадному порядку, використовуючи метод sort() із порівнянням, що забезпечує коректне ранжування гравців за їхніми досягненнями.

Серверна частина в saveRating.php обробляє запити на збереження результатів ігор, декодуючи JSON-запит із полями gamefile, userName, rate і time, валідуючи дані, додаючи запис до ratings['games'][$gameFile] у ratings.json, перезаписуючи файл через file_put_contents, повертаючи JSON-відповідь { status: 'success' }` або `{ status: 'error', message: '...' } , із перевіркою розміру файлу і логуванням операцій для відстеження змін.

Приклад коду ratingViewer.html демонструє повну реалізацію клієнтської логіки, включаючи випадаючий список ігор, таблицю рейтингів і обробку помилок, із асинхронними запитами через fetch для завантаження даних і динамічним оновленням UI, що забезпечує зручність і ефективність роботи.

3.5 Розробка програмного модуля даних

Розробка програмного модуля даних вебсистеми для розв’язування логічних забезпечила централізоване сховище через файли params.json і ratings.json, із серверною логікою в saveGame.php і saveRating.php. Цей модуль підтримує операції читання і запису метаданих ігор та рейтингів, із валідацією даних, обробкою помилок і контролем розміру файлів, що гарантує надійність і сумісність із модулями ігор, адміністрування та рейтингів.

Функція readData(filePath) у saveGame.php і saveRating.php забезпечує читання JSON-файлів для доступу до даних. Логіка передбачає використання file_get_contents() для зчитування вмісту файлу filePath params.json або ratings.json, декодування JSON через json_decode() у масив, із ініціалізацією

порожньої структури (['games' => []] для params.json або ['games' => []] для ratings.json), якщо файл відсутній, що забезпечує коректну роботу при першому запуску. При помилці читання (наприклад, файл пошкоджений) повертається порожній масив, а помилка логуються через error_log() для подальшого аналізу.

Функція writeData(filePath, data) у saveGame.php і saveRating.php відповідає за запис даних у JSON-файли. Логіка включає перевірку розміру даних обмеження до 1 МБ для запобігання перевантаження, кодування масиву data у JSON через json_encode() із прапорцем JSON_PRETTY_PRINT для читабельності, і запис у файл через file_put_contents(). Перед записом перевіряється доступність файлу на запис, із логуванням помилок через error_log() у разі невдачі, що забезпечує надійність збереження.

Функція validateGameData(input) у saveGame.php валідує дані для операцій додавання, редагування або видалення ігор у params.json. Логіка передбачає перевірку наявності полів gameLabel і gamefile для додавання/редагування або видалення, формат gamefile закінчується на ".html", та має лише латинські символи, довжини gameLabel має 1-50 символів, із поверненням помилки {status: 'error', message: 'Некоректні дані'}, якщо перевірка не пройдена, що запобігає запису некоректних даних.

Функція validateRatingData(input) у saveRating.php валідує дані для збереження рейтингів у ratings.json. Логіка включає перевірку наявності полів gamefile, userName, rate і time, із додатковими умовами: userName від 1 до 20 символів, які латинські символи, rate має значення від 0 до 100, time не має бути від’ємне, із поверненням помилки { status: 'error', message: 'Некоректні дані' } у разі невідповідності, що гарантує цілісність даних.

Функція updateGameList(action, input, games) у saveGame.php обробляє запити на оновлення params.json. Логіка залежить від параметра action: для add додається новий запис { gameLabel, gamefile } до масиву games, для update оновлюється запис за gamefile, для delete видаляється запис за gamefile, із

перевіркою існування запису перед операцією, що забезпечує коректне оновлення списку ігор.

Функція addRating(gamefile, rating, ratings) у saveRating.php додає новий запис рейтингу до ratings.json. Логіка передбачає ініціалізацію масиву ratings['games'][$gamefile], якщо він відсутній, додавання запису { userName, rate, time } до цього масиву, із подальшим перезаписом файлу через writeData(), що забезпечує збереження результатів ігор.

Функція logOperation(operation, details) у saveGame.php і saveRating.php логує операції для відстеження змін. Логіка включає запис інформації про операцію у файл логів через error_log(), із додаванням мітки часу, що полегшує діагностику проблем. На рисунку 3.10 зображено приклад логів збережених на сервері під час роботи вебсистеми

Рисунок 3.10 – Логування вебсистеми

SaveGame.php демонструє повну реалізацію серверної логіки для обробки запитів, із валідацією, оновленням params.json і логуванням, а saveRating.php забезпечує аналогічну функціональність для ratings.json, із підтримкою асинхронних запитів через fetch від інших модулів.

3.6 Розробка інтерфейсу вебсистеми

При розробці інтерфейсу програмного забезпечення було приділено особливу увагу його зрозумілості та зручності для користувача. Реалізовано чорно-білий/схематичний дизайн інтерфесу який буде мати подальшу розробку у кольоровій формі далі представлені макети.

Головне меню додатку «Бик Ейнштейна» – буде першим етапом взаємодії з додатком користувач потрапляє на головний екран, де представлено інтуїтивне меню з чотирма основними розділами. В центрі уваги – персоналізація: доступ до гри надається після введення імені. Такий підхід дозволяє реалізувати фіксацію результатів конкретного користувача та забезпечує індивідуальний досвід(рис.3.11).

Рисунок 3.11 – Схематичний дизайн головного меню додатку

При натисканні кнопки «Бики та Корови» має відкритись інтерфейс гри «Бики та Корови» (рис. 3.12). Інтерфейс буде демонструвати логіку гри після натискання відповідної кнопки в меню. Користувач обирає рівень складності,

вводить числову комбінацію для вгадування та взаємодіє з кнопками «Вгадати» або «Здатися». Додаткові опції дозволяють повернутися до головного меню. Система підрахунку спроб створює відчуття прогресу та змагання, а контекстно-залежні інструкції роблять взаємодію прозорою.

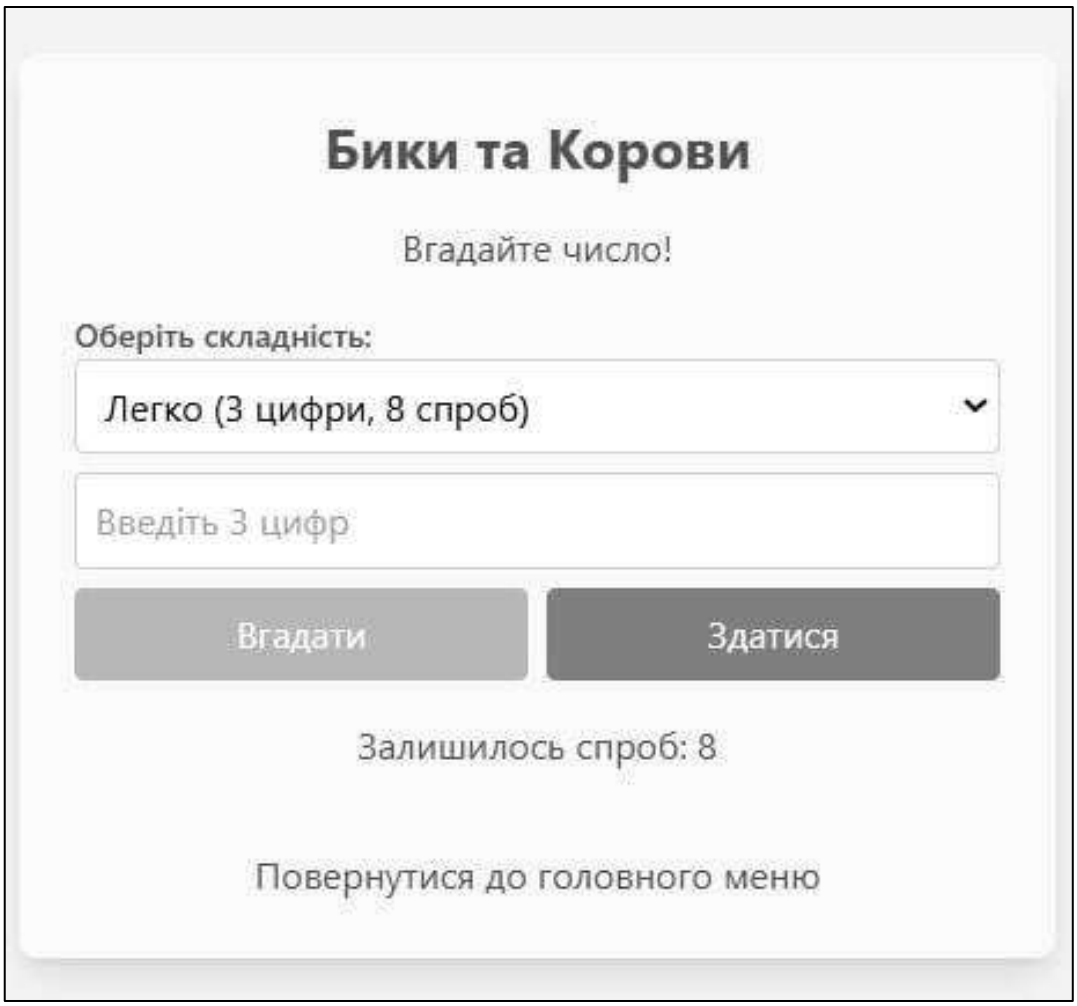


Рисунок 3.12 – Схематичний гри «Бики та Корови»

При натисканні кнопки «Задача Енштейна» на рисунку 3.13 зображено макет інтерфейсу, який має відкритись для іншого логічного завдання із запитанням «Хто володіє зеброю?» – розроблено окремий інтерфейс з таблицею для заповнення атрибутів. Користувач поступово розв’язує логічну задачу, обираючи будинки та атрибути, які перевіряються на правильність на основі підказок . вводить пропозицію де знаходиться зебра та взаємодіє з

кнопками «Перевірити» або «Здатися». Додаткові опції дозволяють повернутися до головного меню.

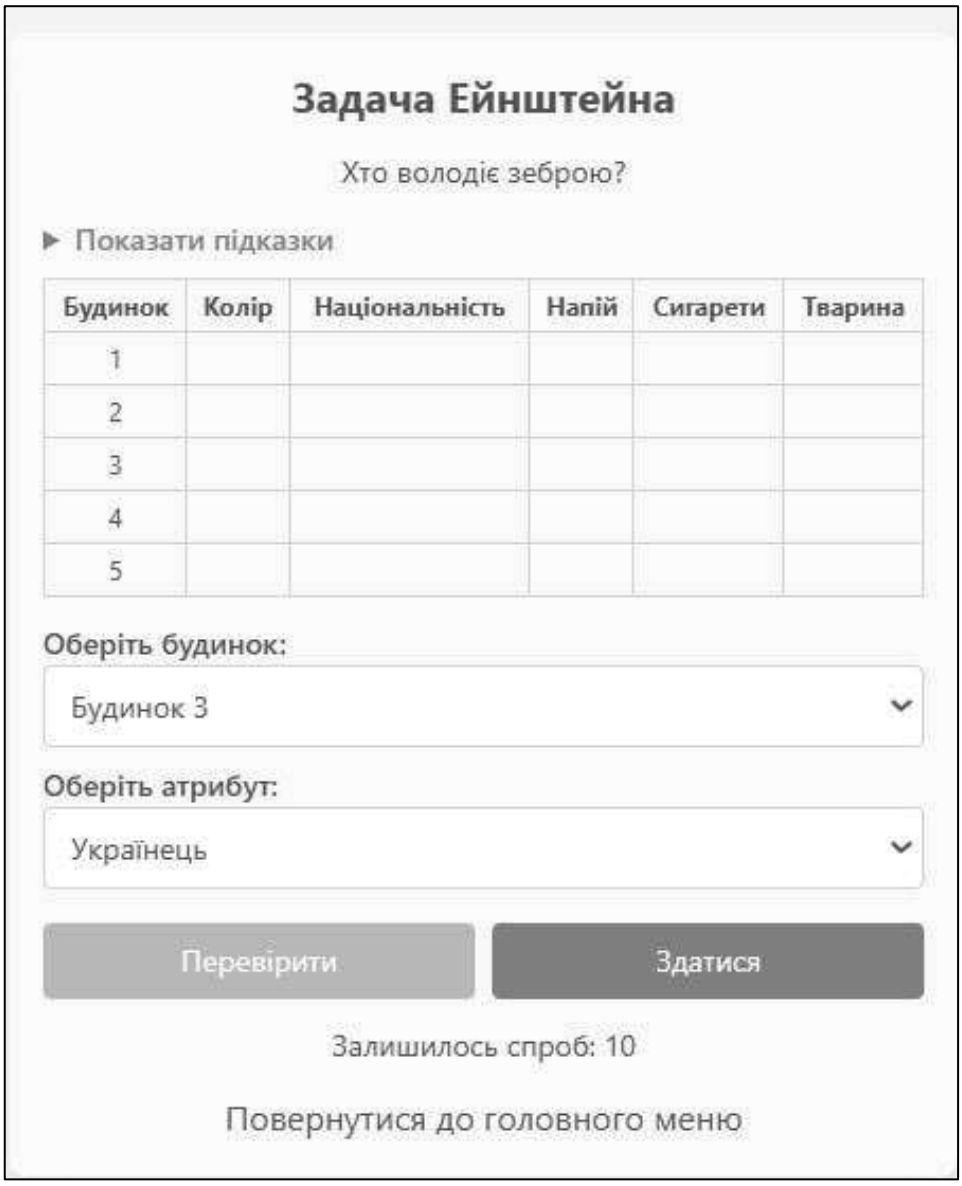


Рисунок 3.13 – Схематичний дизайн гри «Задача Енштейна»

На рисунку 3.14 зображено схематичне подання того що відбувається при натисканні кнопки «подивитись весь рейтинг», тобто інтерфейс який має використовуватись. Є скролер в якому обирається гра для перегляду рейтингу. Створено два стовпця з інформацією, один не відсортований інший відсортаваний по критеріям очків також з інформації зафіксовано ім'я

користувача та час проходження. Розроблена можливість завжди вийти в головне меню при нажатю кнопки «повернутись до головного меню».

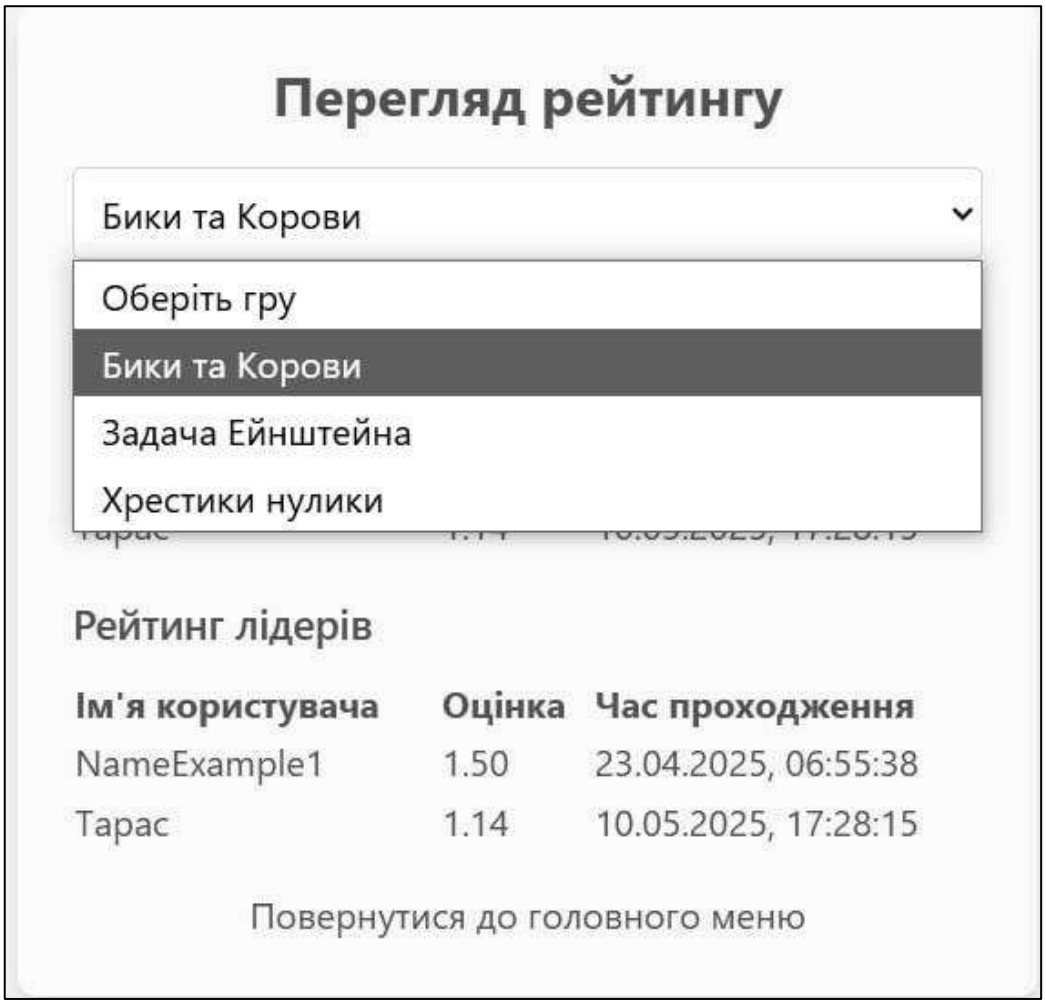


Рисунок 3.14 – Схематичний дизайн модуля рейтингу

Інтерфейс адміністративної панелі вебсистеми для розв’язання логічних задач готовий до використання, пропонуючи зручний доступ до керування іграми через файл adminPanel.html. Інтерфейс складається з інтуїтивних елементів для входу, перегляду списку ігор, додавання нових ігор, редагування наявних і видалення, із плавними оновленнями.

На рисунку 3.15 зображено схематичне подання форми автентифікації, розташоване посередині екрана, де присутнє поле passwordInput для введення пароля з підказкою «Введіть пароль» і кнопка «Увійти». Після успішного

введення правильного пароля форма зникає, відкриваючи головний екран для роботи з іграми.

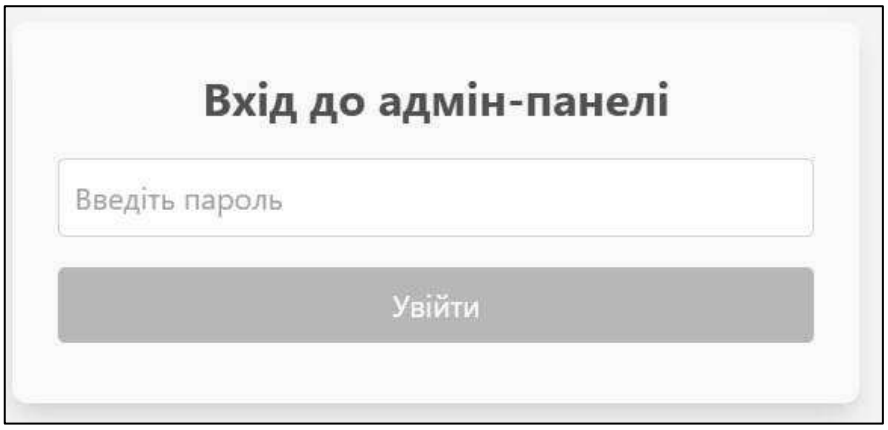


Рисунок 3.15 – Схематичний дизайн форми автентифікації адміністративної панелі

На рисунку 3.16 зображено схематичне подання головного екрана, де відображається список ігор gamesList у вигляді карток із назвами gameLabel, наприклад, «Бики та Корови», поряд із кнопками «Редагувати» і «Видалити», відсортованими за алфавітом. У нижній частині розташована кнопка «Додати нову гру», яка публікує новий тип логічних завдань.

Рисунок 3.16 – Схематичний дизайн головної форми адміністративної панелі

При розробці інтерфейсу приділялася більша увага користувацькому досвіду, а не красі. А на етапі розробки були додані кольори, для кращогої естетики та спокійної гами кольорів. Загалом, інтерфейс розроблено з акцентом на інтуїтивність, зворотний зв’язок і логічну послідовність взаємодій. Кожен елемент має чітку функцію й візуально витримано в одному стилі принципам сучасного UX-дизайну

3.7 Висновки

У третьому розділі було обґрунтовано вибір технологій та інструментів, що використовувались для розробки програмних модулів. Виконавши аналіз програмного продукту та його задач було обрано комбінацію мов програмування (HTML/CSS/JS + PHP + JSON).

У ході розробки програмних модулів вебсистеми для розв’язання логічних задач реалізовано комплексну багаторівневу структуру, що включає модулі ігор, адміністрування, рейтингів, даних та користувацький інтерфейс. Усі компоненти створено з урахуванням принципів модульності, масштабованості, зручності використання.

Модуль ігор було спроєктовано так, щоб підтримувати як наявні задачі «Бики та Корови» та «Задача Ейнштейна», так і майбутні, з уніфікованою структурою HTML-елементів і функціональністю. Впроваджено спільні логічні функції, а також унікальні методи для кожної гри. Такий підхід дозволяє гнучко додавати нові завдання без зміни архітектури.

Модуль адміністрування забезпечує повний цикл керування іграми: додавання, редагування, видалення та автентифікацію адміністратора. Реалізована перевірка коректності даних, дублювання, обробка помилок і логування, що підвищує надійність і безпеку роботи з контентом системи.

Модуль рейтингів відповідає за збереження результатів ігор, їх динамічне відображення та сортування. Забезпечено інтуїтивний інтерфейс, автоматичне оновлення інформації через асинхронні запити та валідацію вхідних даних, що робить систему зручною та прозорою для користувачів.

Модуль даних реалізовано через JSON-файли params.json і ratings.json, що використовуються як легка, гнучка й незалежна альтернатива повноцінній базі даних. Функції серверної частини відповідають за коректне читання, перевірку та запис даних, забезпечуючи надійність і відсутність дублювань.

Інтерфейс вебсистеми створено згідно з принципами UX-дизайну: зручність навігації, адаптивність, інтуїтивність та єдність стилю. Реалізовані окремі інтерфейси для гри, головного меню, рейтингу та адміністративної панелі. Особлива увага приділена персоналізації взаємодії через іменування користувачів і гейміфікацію рейтингу.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Тестування програмнних модулів вебсистеми

На рисунку 4.1 відображено стартовий інтерфейс додатку, що містить заголовок, поле для введення імені користувача та чотири основні навігаційні кнопки: «Бики та Корови», «Задача Ейнштейна», «Хрестики нулики» «Зайти в адміністративну панель» та «Подивитись весь рейтинг». Така структура забезпечує інтуїтивну навігацію та акцентує увагу на персоналізованому досвіді – доступ до активних режимів можливий лише після введення імені.

Рисунок 4.2 демонструє механізм валідації введення. При спробі перейти до гри без вказаного імені з'являється модальне повідомлення з підказкою: «Будь ласка, введіть ім'я!». Такий підхід підвищує якість взаємодії, попереджає помилки користувача та підкреслює важливість персоніфікованого підходу.

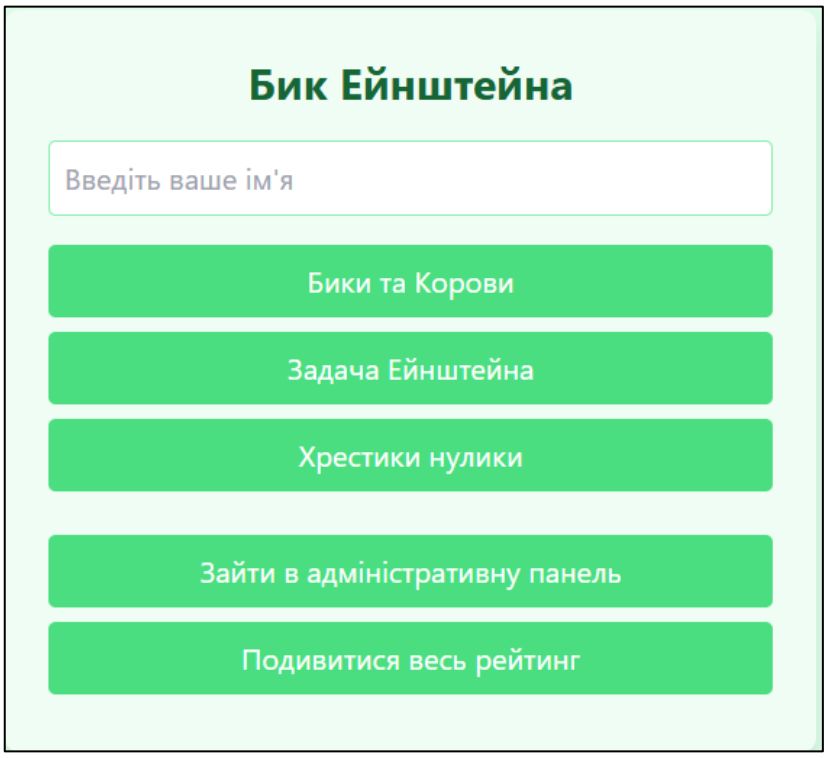


Рисунок 4.1 – Головне меню додатка «Бик Ейнштейна»

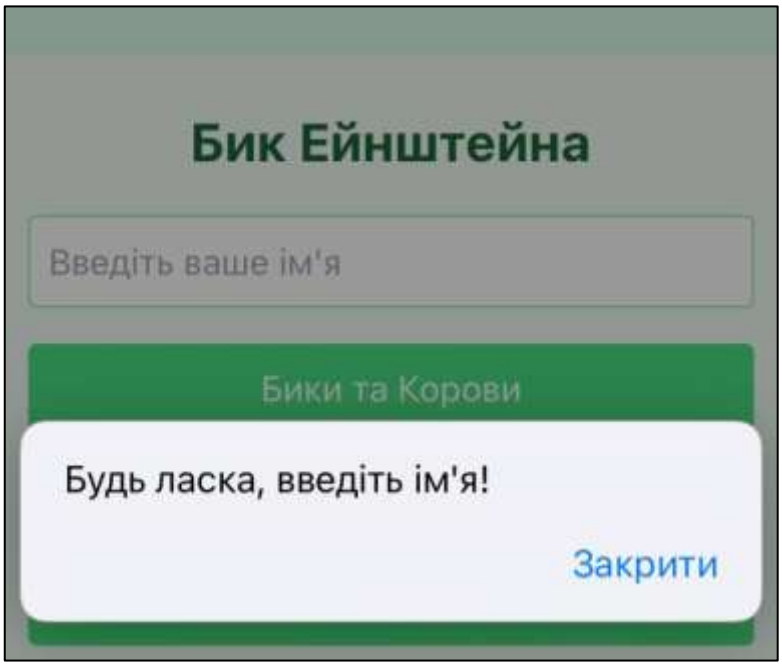


Рисунок 4.2 – Повідомлення про помилку при незаповненому полі імені

Рисунок 4.3 зображає зміни після введення імені наприклад, «Тарас» користувач отримує доступ до ігрового функціоналу. Збереження введеного значення в полі дозволяє легко продовжити взаємодію та покращує UX завдяки відсутності повторного введення.

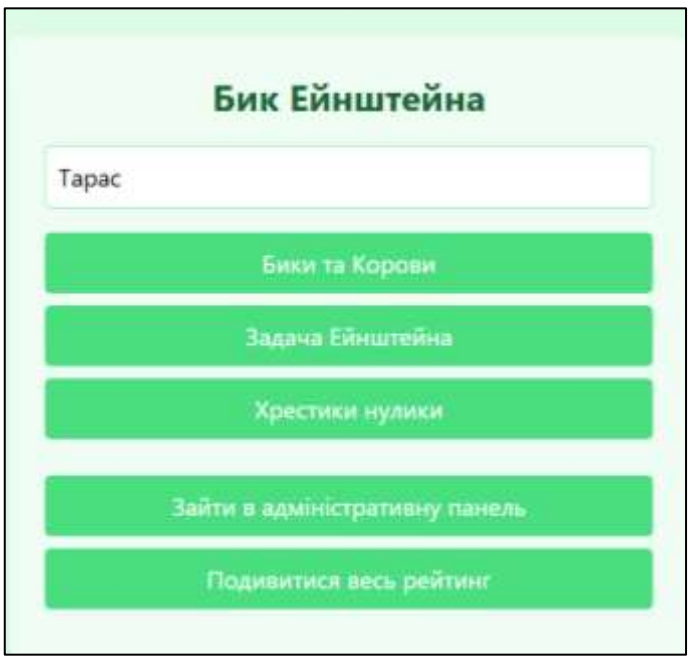


Рисунок 4.3 – Введене ім’я користувача

Рисунок 4.4 демонструє логіку гри після натискання відповідної кнопки в меню. Користувач обирає рівень складності, вводить числову комбінацію для вгадування та взаємодіє з кнопками «Вгадати» або «Здатися». Додаткові опції повернутися до головного меню. Система підрахунку спроб створює відчуття прогресу та змагання, а контекстно-залежні інструкції роблять взаємодію прозорою.

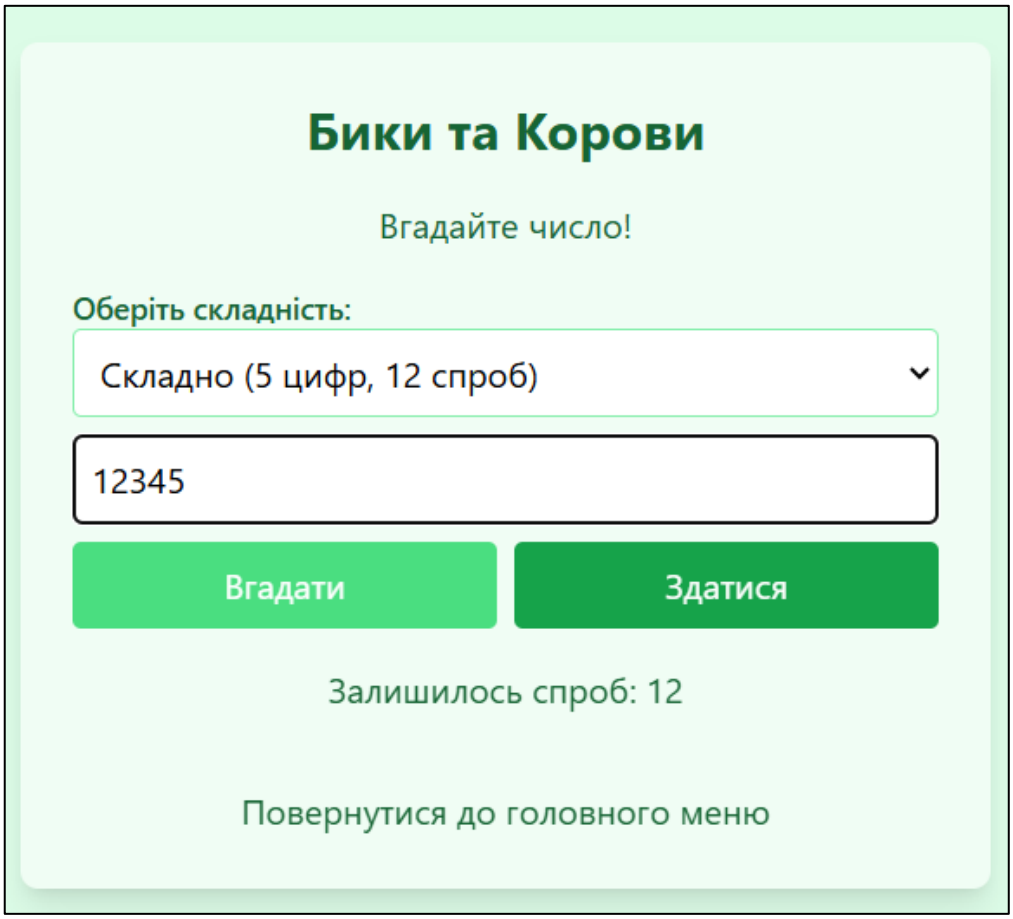


Рисунок 4.4 – Інтерфейс гри «Бики та Корови»

На рисунку 4.5 користувач встановив рівень складності як «Середньо», що передбачає 4-значне число й 10 спроб. Підказка під полем введення додатково інформує про необхідність вводити 4 різні цифри, що зменшує ймовірність помилки й підтримує правильну логіку гри. Якщо умову в підсказці не виконати при спробі вгадати число видасть помилку.

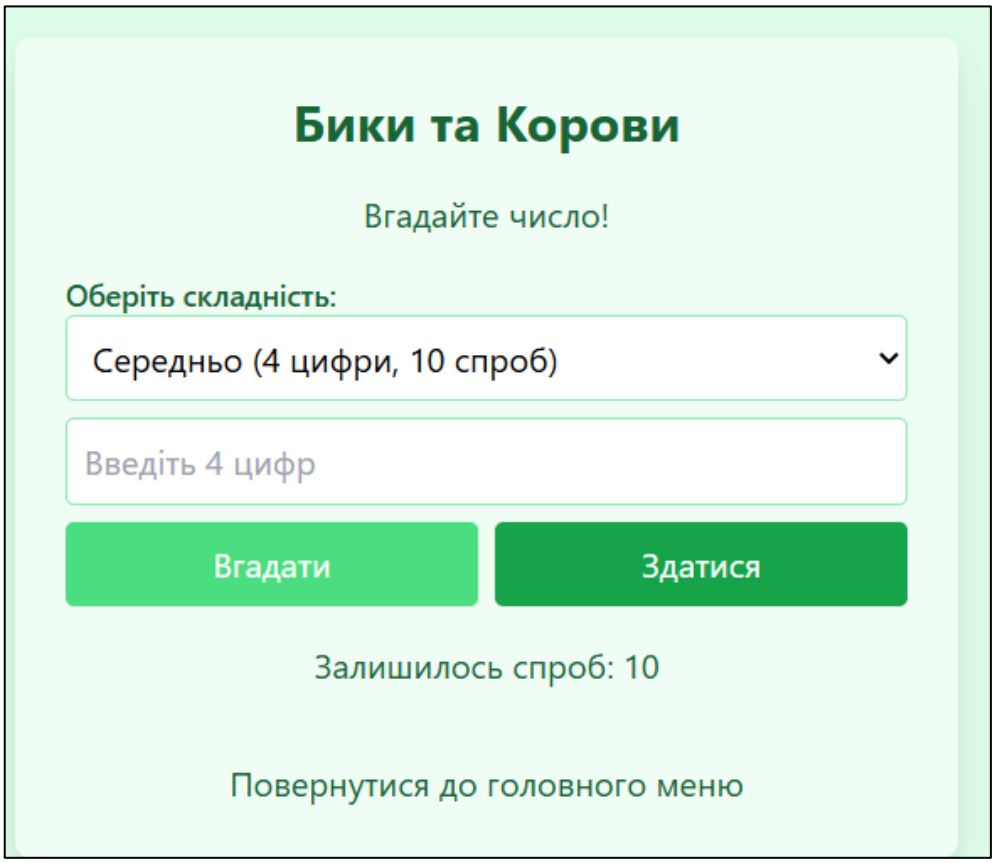


Рисунок 4.5 – Обрано середній рівень складності гри

На рисунку 4.6 користувач ввів комбінацію «1234» та підготувався до перевірки. Індикатори кількості спроб та інструкція залишаються активними, підсилюючи користувацький контроль і збереження контексту.

На рисунку 4.7 після натискання кнопки «Вгадати» зображено, як система відповідає: 0 биків, 4 корови, що означає – всі цифри правильні, але не на своїх місцях. Нижче формується історія спроб: відображено комбінацію та результат у скороченому форматі «1234: 0Б 4К».

Рисунок 4.8 демонструє розвиток гри: користувач здійснив три спроби (1234, 5678, 9012), з кожною з яких система фіксує результат. Історія спроб відображається у форматі, зручному для порівняння, що дозволяє користувачеві здійснювати дедуктивний аналіз і наближатися до правильного варіанту.

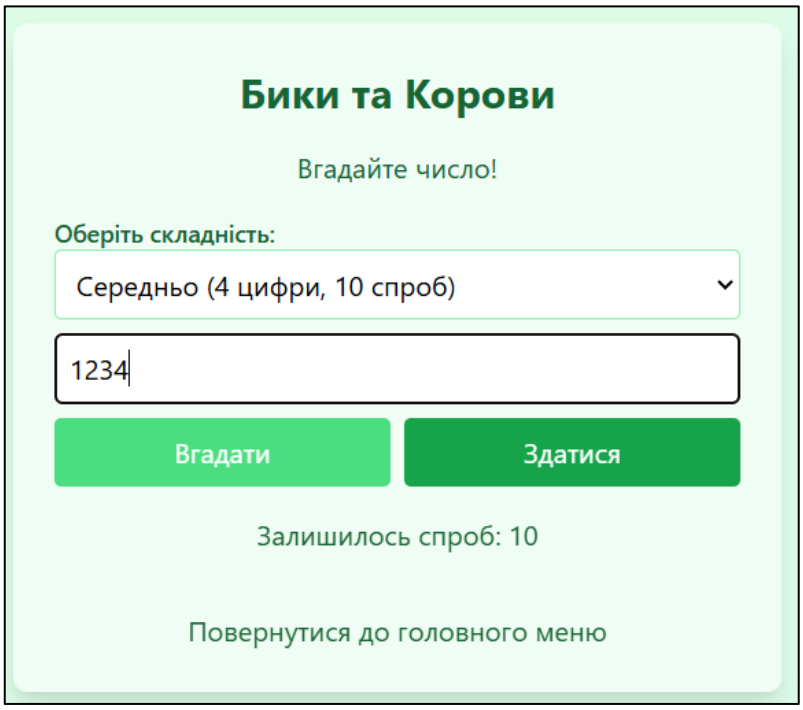


Рисунок 4.6 – Приклад першої спроби введення числа

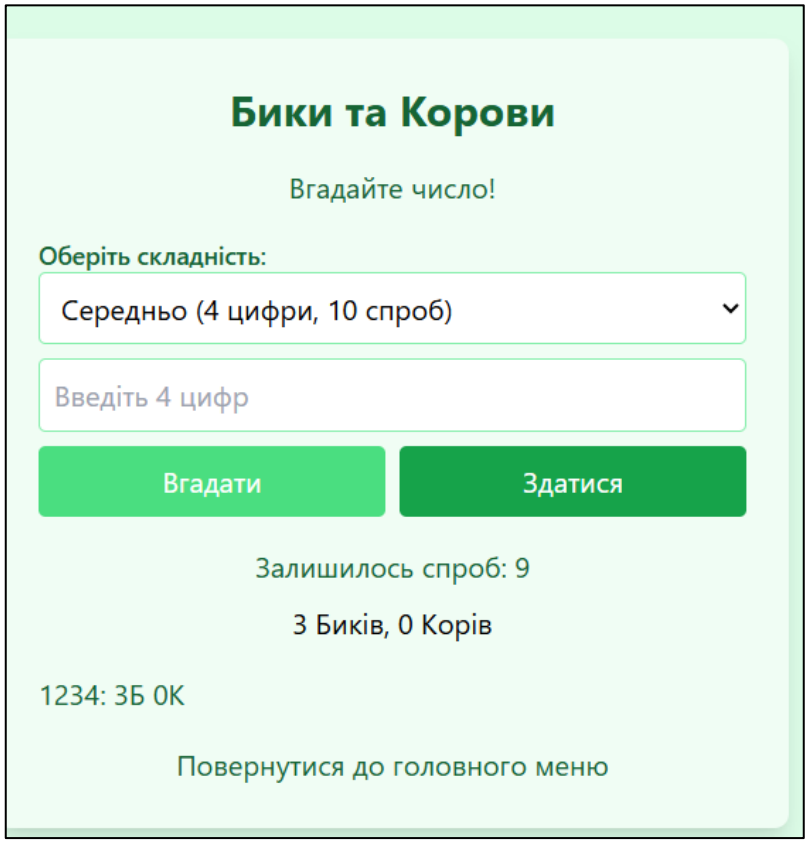


Рисунок 4.7 – Відповідь системи після першої спроби

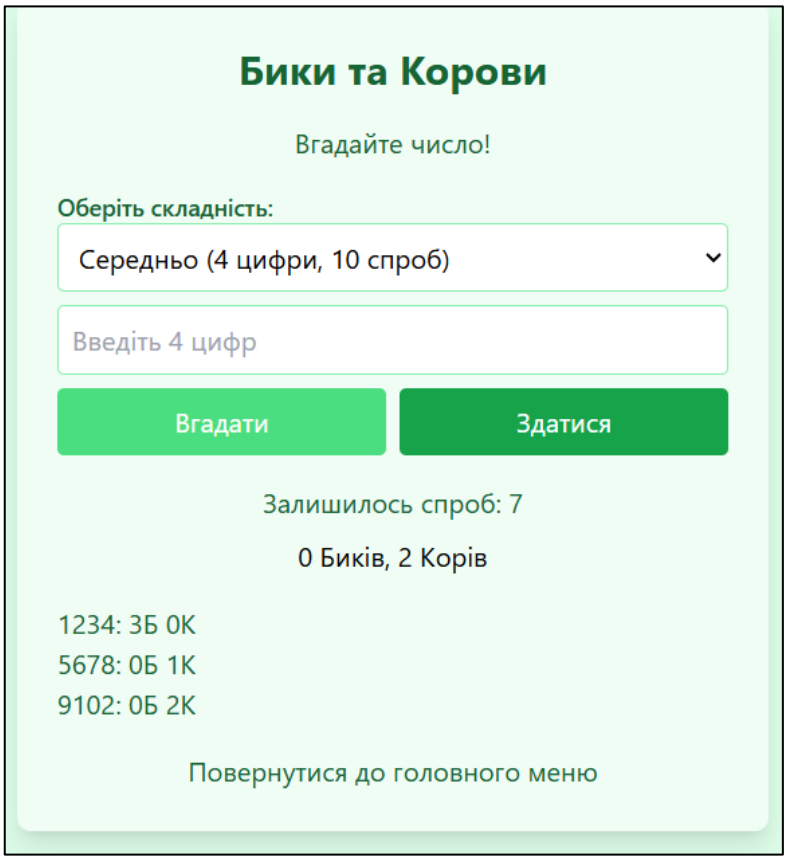


Рисунок 4.8 – Продовження гри з накопиченням історії спроб

Цей екран відображається, коли користувач не вгадав число протягом відведених 10 спроб(рис. 4.9). Система повідомляє: «Гра закінчена! Ви вичерпали спроби. Число було 1237». Нижче наведено повну історію введених комбінацій та відповідей, що дозволяє користувачеві проаналізувати хід гри та зробити висновки для наступної спроби.

Рисунок 4.10 демонструє сценарій при якому користувач вгадав число 7594 за 6 спроб, і система надає вітальне повідомлення разом з історією ходів. Така візуалізація створює відчуття досягнення, а також дозволяє користувачеві відслідковувати свою ефективність та логіку міркувань упродовж гри.

Історія спроб, що відображається у вигляді структурованої таблиці із зазначенням кількості «биків» і «корів» для кожної введеної комбінації, сприяє розвитку аналітичного мислення гравця, оскільки допомагає аналізувати

власні припущення та вибудовувати стратегію подальших дій на основі отриманих підказок.

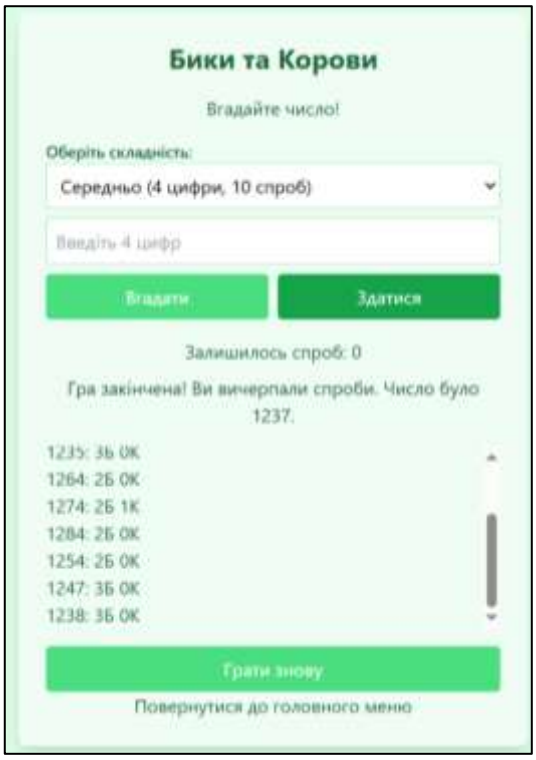


Рисунок 4.9 – Повідомлення про завершення гри після вичерпання спроб

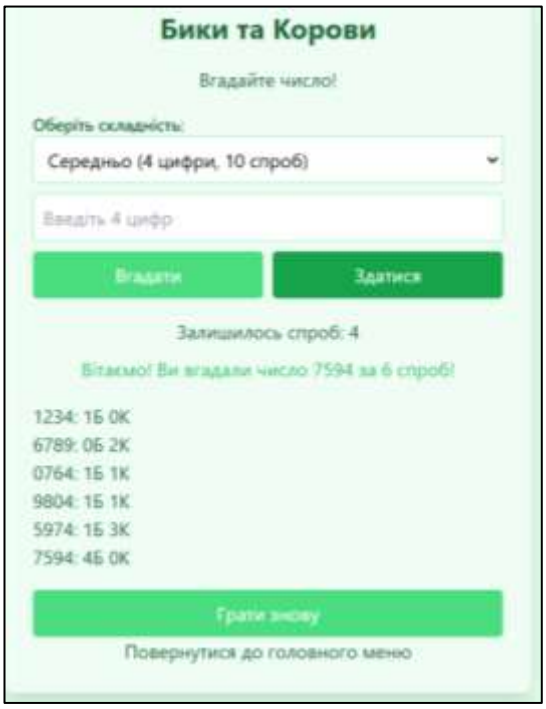


Рисунок 4.10 – Повідомлення про успішне завершення гри

Рисунок 4.11 демонструє початковий вигляд логічної задачі «Хто володіє зеброю?». У верхній частині – заголовок і запитання. Нижче розташована таблиця з п’ятьма будинками та категоріями атрибутів (колір, національність, напій, сигарети, тварина). Користувач обирає будинок і атрибут, після чого має змогу перевірити свою гіпотезу або здатися. Такий формат забезпечує максимальну зручність для побудови логічних зв’язків.

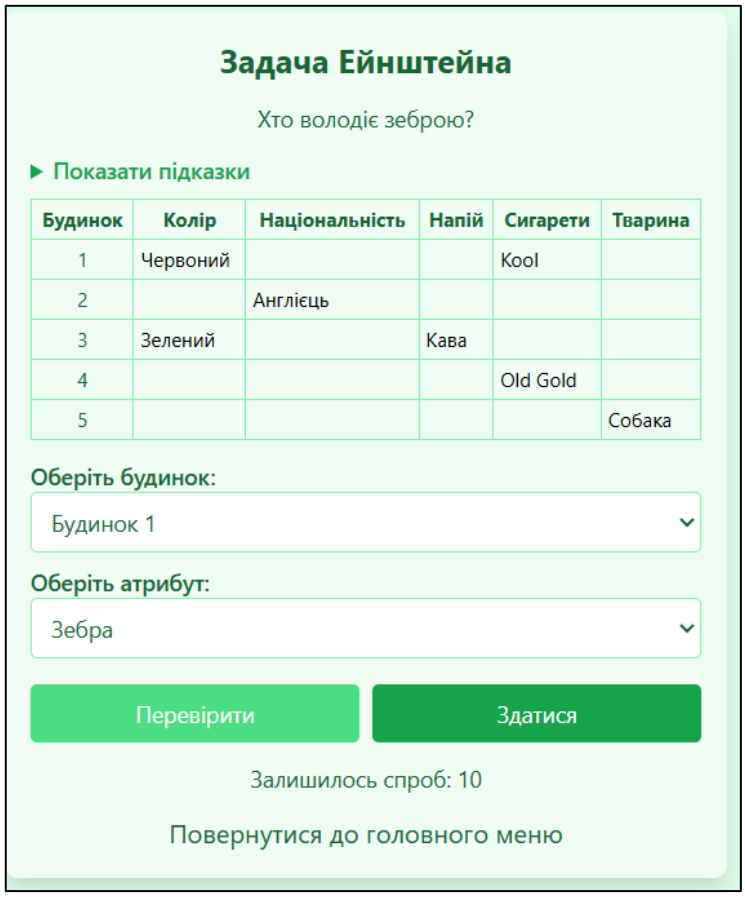


Рисунок 4.11 – Інтерфейс «Задачі Ейнштейна»

На рисунку 4.12 активовано опцію «Показати підказки». Підказки подано у вигляді нумерованого списку з чітко позначеними категоріями (національність, напій, сигарети, тварина тощо), що дозволяє користувачеві застосовувати дедуктивний підхід для заповнення таблиці. Це підвищує когнітивну привабливість гри й підтримує логічне мислення.

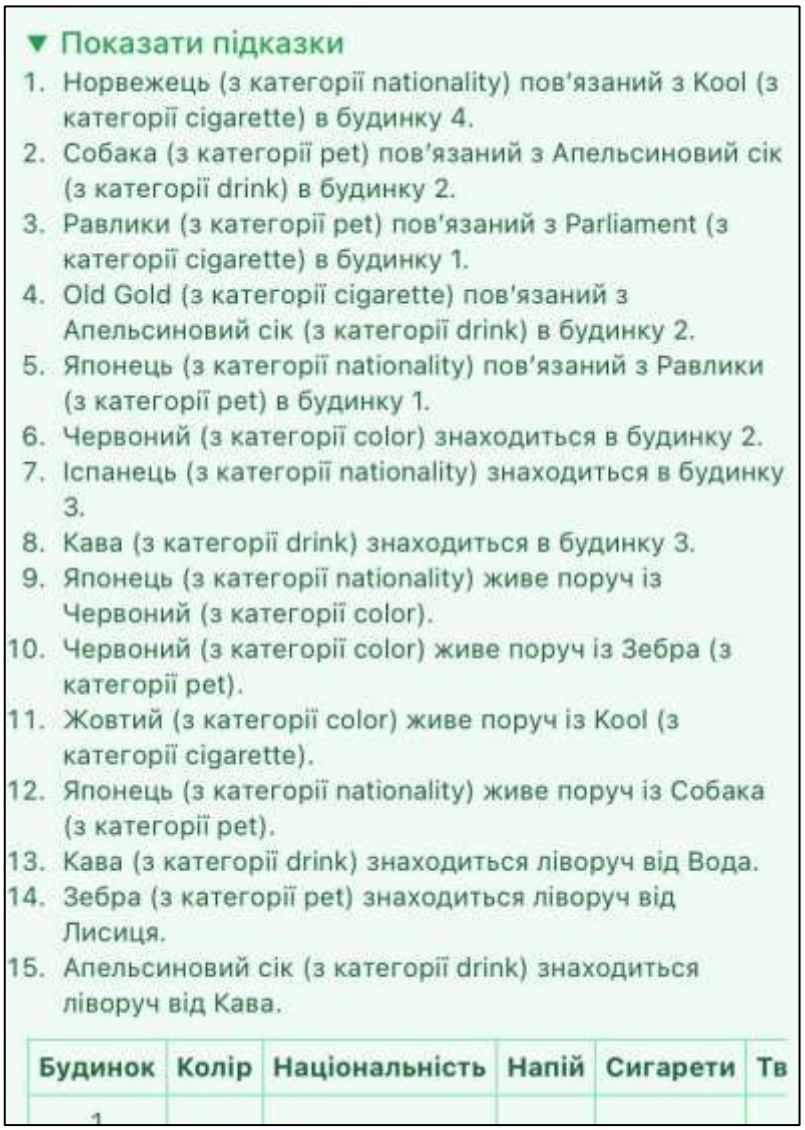


Рисунок 4.12 – Відображення підказок у «Задачі Ейнштейна»

Результат зображений на рисунку 4.13 з'являється після останньої (десятої) невдалої спроби. Система повідомляє, що зебра була в будинку 3. Спроби витрачаються при перевірці наявності атрибутів в будинках як і самої зебри, яку потрібно знайти. Повідомлення подано чітко й з дотриманням UX-принципів: користувач одразу бачить результат і має змогу обрати подальші дії – почати гру спочатку або повернутись до меню.

На рисунку 4.14 продемонстровано сценарій, коли користувач свідомо здається, не витративши жодної спроби. Після натискання кнопки «Здатися» система відображає правильну відповідь – зебра була в будинку 2. Завдяки

цьому користувач отримує зворотний зв'язок і може проаналізувати свою логіку на майбутнє.

Рисунок 4.13 – Повідомлення про програш після вичерпання спроб у «Задачі Ейнштейна»

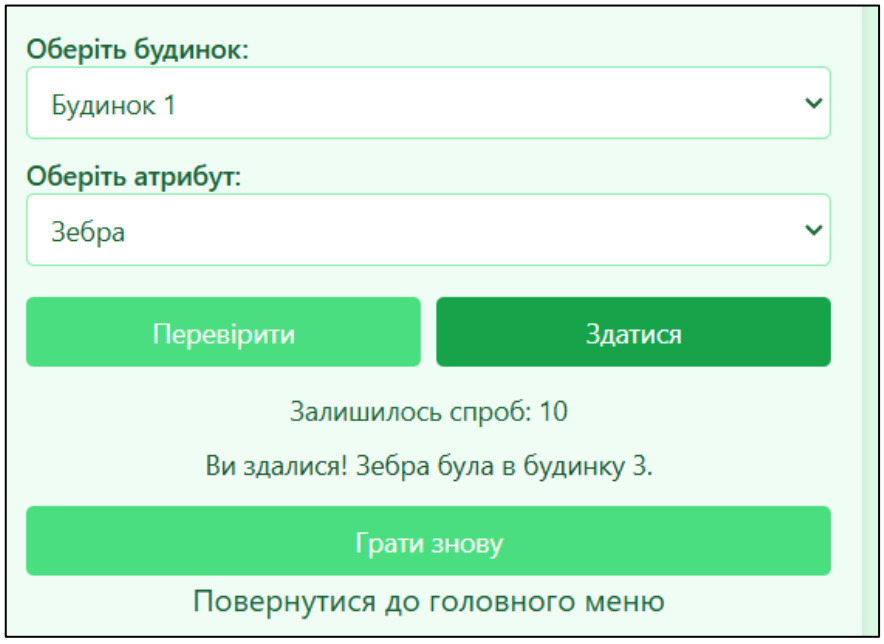


Рисунок 4.14 – Повідомлення про капітуляцію гравця

Важливий для тестування є модуль рейтингу на рисунку 4.15 продемонстровано головний екран вибору для якої гри подивитись рейтинг.

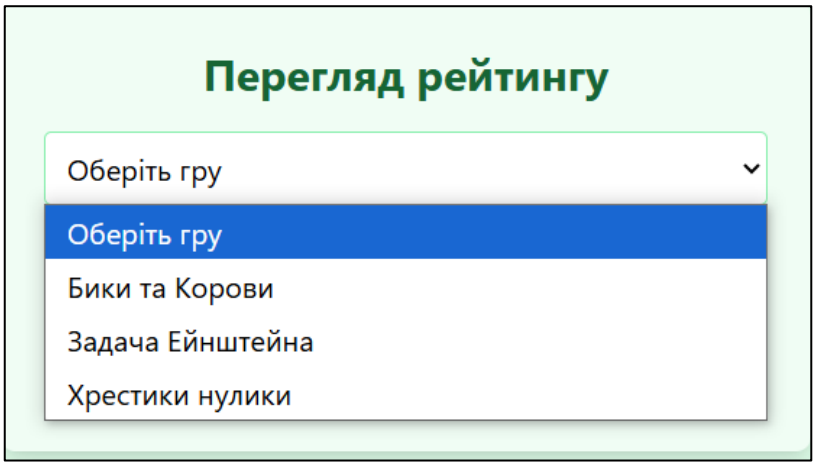


Рисунок 4.15 – Перегляд рейтингу для обраної гри

На рисунку 4.16 продемонстровано приклад надання рейтингу для задачі «Бики та Корови» На формі зображено ім’я користувача час проходження та рейтинг отриманий за проходження задачі також є таблиця лідерів де користувачів відсортовано за рейтингом.

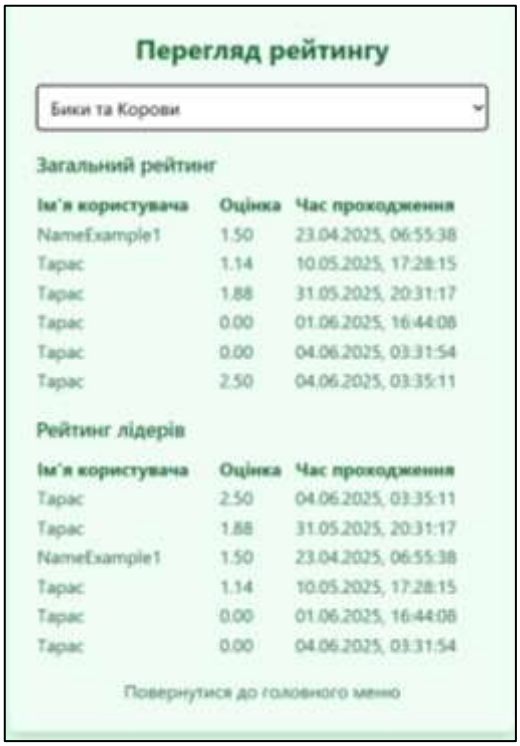


Рисунок 4.16 – Перегляд рейтингу для «Бики та Корови»

На рисунку 4.17 продемонстровано приклад надання рейтингу для «Задачі Ейнштейна».

Рисунок 4.17 – Перегляд рейтингу для «Задачі Ейнштейна»

Також важливим є тестування модуля адміністрування. На 4.18 надано приклад помилки вводу пароля для авторизації адміністратора щоб увійти в адміністративну панель.

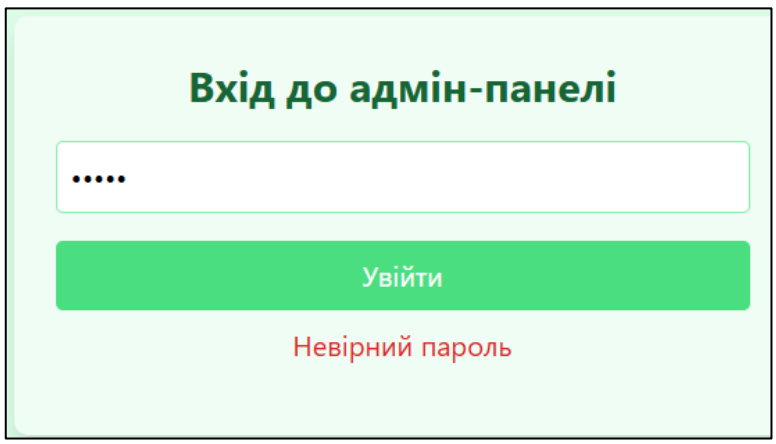


Рисунок 4.18 – Помилка авторизації адміністратора

Після успішної авторизації адміністратора отримавши доступ до панелі адміністрування є можливість виконати 3 типи дій. Додавання нового типу логічних задач (рис.4.19), редагування коду наявних логічних задачах на пряму через адміністративну панель (рис.4.20) та видалення гри з вебсистеми для розв’язання логічних завдань(рис.4.21).

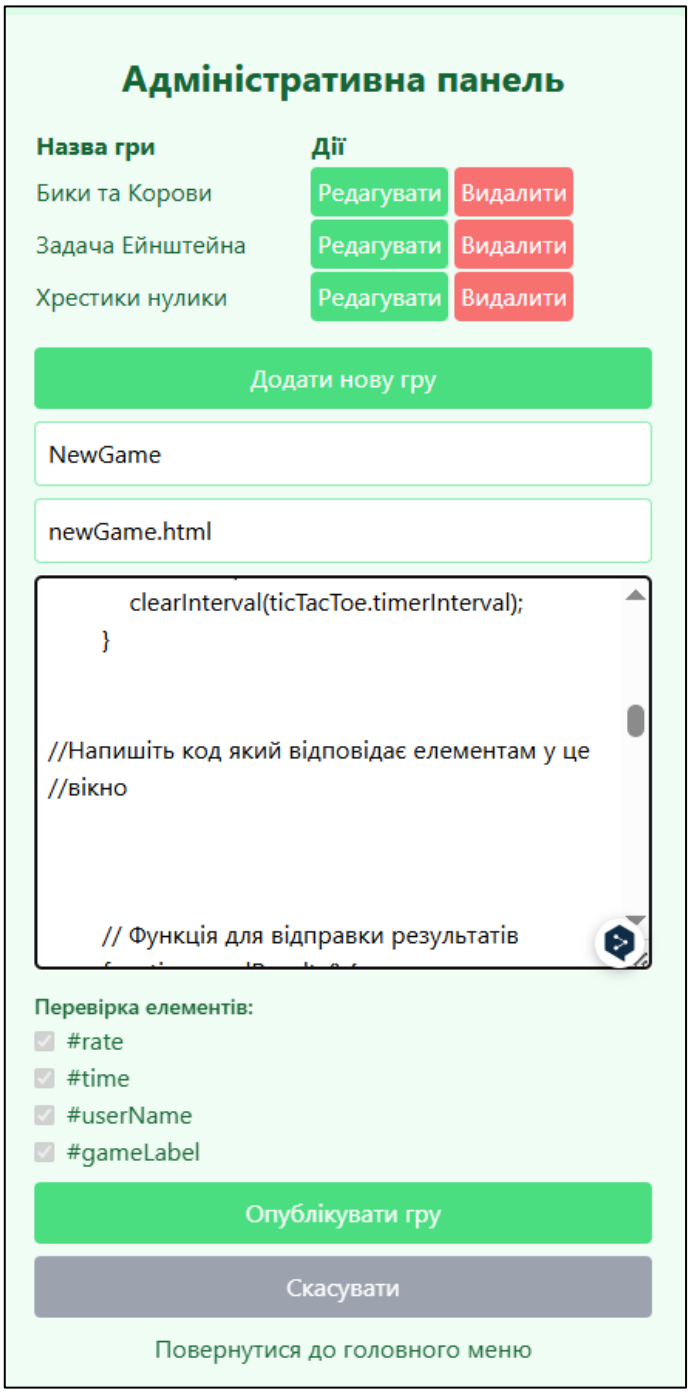


Рисунок 4.19 – Додавання нової логічної задачі у вебсистему

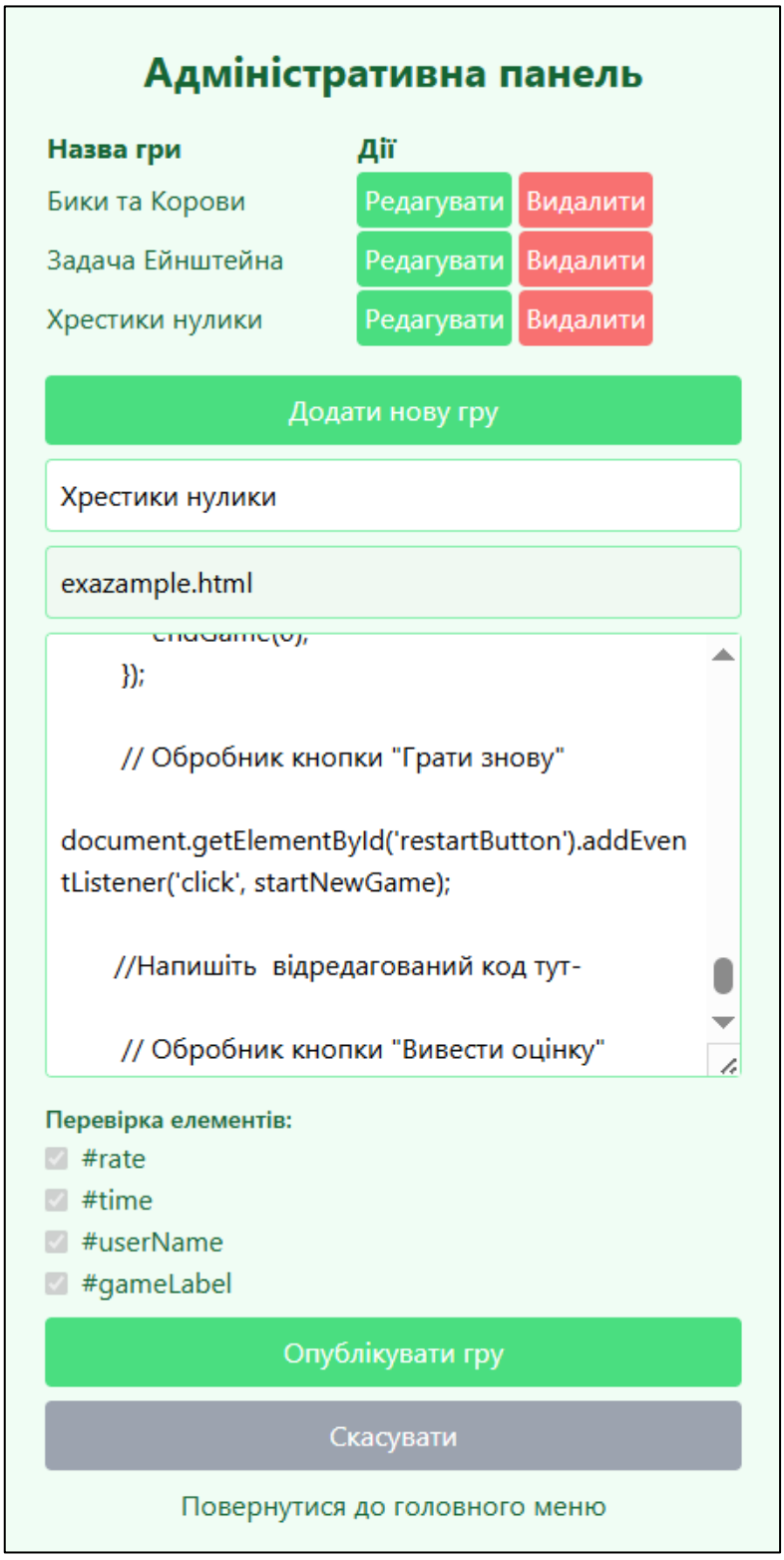


Рисунок 4.20 – Редагування кода існуючої задачі

Після перевірки необхідних елементів та натискання кнопки «Опублікувати гру» зміни вступають в силу та надійде повідомлення про успішне редагування або додавання задачі.

Рисунок 4.21 – Видалення існуючої гри

Після публікації нову задачу додано до вебсистеми але наступним кроком одразу видалено після повідомлення про успіх. Зміни у списку наявних завдань відображено на рисунку 4.22.

Рисунок 4.22 – Повернення до стартового стану списку задач

Протестований повний цикл роботи з вебсистемою адміністратором та отримано очікуванні результати.

4.2 Розробка інструкції користувача

Інструкція користувача передбачає визначення технічних вимог для запуску програмного продукту. Деталі щодо мінімальної та рекомендованої конфігурації пристрою для запуску вебсистеми можна знайти в таблицях 4.1 та 4.2.

Таблиця 4.1 – Мінімальна конфігурація:

Підключення до мережі:	512 Кбіт/с
Браузер	Google Chrome 70+, Mozilla Firefox 65+, Microsoft Edge 80+, Safari 12+
Екран	мінімальна роздільна здатність 1024×768
Об’єм оперативної пам’яті	2 ГБ для 64-разрядної системи

Таблиця 4.2 – Рекомендована конфігурація:

Підключення до мережі:	від 5 Мбіт/с
Браузер	Google Chrome 70+, Mozilla Firefox 65+, Microsoft Edge 80+, Safari 12+
Екран	Full HD (1920×1080) або вища
Об’єм оперативної пам’яті	4 ГБ для 64-разрядної системи

Після відкриття вебсайту користувач бачить головний екран із формою для введення свого імені. Користувач вводить своє ім'я у відповідне текстове поле та натискає кнопку початку гри. Далі користувачеві пропонується обрати одну з доступних логічних задач: «Бики та Корови» або «Задача Ейнштейна» або інше завдання доданне адміністратором.

У разі вибору гри «Бики та Корови», користувач додатково обирає рівень складності, який визначається кількістю цифр у секретному числі (наприклад, 3, 4 або 5 цифр). Після початку гри користувач вводить свої припущення щодо секретного числа у спеціальне поле та надсилає спробу натисканням кнопки. Після кожної спроби система надає підказку у вигляді кількості «биків» і «корів», що допомагає користувачу скоригувати наступні припущення. Процес повторюється, доки користувач або не вгадає правильне число, або не вичерпає ліміт спроб.

У разі вибору «Задачі Ейнштейна», користувач отримує початковий набір підказок і має за допомогою логічних висновків встановити відповідності між характеристиками об'єктів (наприклад, кольором будинку, національністю власника, видом напою тощо). Введення рішення здійснюється через форму або таблицю, а перевірка правильності відбувається автоматично після заповнення всіх обов'язкових полів.

Після завершення гри за одним із сценаріїв виграш, програш або здача система фіксує результат. На основі кількості зроблених спроб, часу розв’язання та результату проходження формується рейтинг. Дані автоматично передаються на сервер, де зберігаються у форматі JSON для формування особистої історії ігрових сесій і загального рейтингу користувачів.

Користувач також має можливість переглянути історію своїх ігор та рейтинг інших учасників через спеціальне модальне вікно. Там відображаються особисті результати, та результати лідерів. Після перегляду статистики користувач може розпочати нову гру або повернутися на головну сторінку.

Інструкція для адміністратора вебсистеми розв’язування логічних задач створена для забезпечення зручного керування задачами через адміністративну панель. Процес входу у адміністративну панель розпочинається з відкриття браузера і переходу до нажаттям кнопки адміністративної панелі. На екрані з’являється форма автентифікації потрібно

ввести унікальний пароль щоб запобігти входу звичайного користувача у панель.

Після входу доступний перегляд списку ігор доступний у редакторі, де список логічних задач представлений у вигляді карток із назвами. Додавання нової гри виконується через натискання кнопки «Додати нову гру» у нижній частині редактора. Відкривається форма додання із полями для назви задачі до 50 символів і назву для файлу гри має закінчуватися на «.html», та додання коду задачі. Після введення даних, наприклад, назви «Бики та Корови» і файлу «bullsAndCows.html», натискання кнопки «Зберегти» додає гру до списку редактору, якщо немає дублювання, інакше з’являється повідомлення помилки.

Редагування гри можливе через натискання кнопки «Редагувати» у списку редактора. Форма відкривається із заповненим полем gameLabel і заблокованим полем gameFile. Після внесення змін у назву гри натискання «Зберегти зміни» оновлює запис у списку задач, а у разі помилки відображається повідомлення для корекції даних.

Видалення гри здійснюється натисканням кнопки «Видалити» у списку редактора, після чого з’являється спливне вікно для підтвердження дії. Натискання «OK» видаляє гру зі списку, а «Cancel» скасовує операцію. При успішному видаленні список оновлюється.

Рекомендації для адміністратора включають перевірку введених даних перед збереженням, використання унікальних назв файлів для ігор, регулярне оновлення списку для синхронізації із сервером, а також звернення до документації або логів у разі технічних проблем. Технічні зауваження стосуються необхідності локального сервера для роботи із saveGame.php, уникнення ручного редагування params.json без резервної копії та можливого введення пагінації при великій кількості ігор у майбутньому.

4.3 Висновки

У четвертому розділі було проведено всебічне тестування вебсистеми для розв’язування логічних задач. Що підтвердило коректність роботи користувацького інтерфейсу: забезпечено валідацію введення імені, обробку помилок, логіку гри відповідно до вибраного рівня складності та правильне формування підказок у процесі проходження задач.

Модуль «Бики та Корови» продемонстрував стабільну роботу логіки гри, підрахунок спроб, історію введених комбінацій, підказки у вигляді кількості «биків» і «корів» та відображення результату залежно від сценарію (виграш, програш, здача). Аналогічно протестовано модуль задачі «Хто володіє зброю?» – перевірено коректність підказок, введення атрибутів, логіку завершення гри та систему відображення відповідей.

Модуль рейтингів успішно виконує функції сортування, фільтрації та відображення результатів. Користувач має змогу переглянути свій рейтинг, час проходження ігор та порівняти результати з іншими гравцями.

Тестування адміністративної панелі підтвердило можливість додавання, редагування та видалення логічних задач із перевіркою введених даних. Реалізована система повідомлень дозволяє вчасно попереджати про помилки або підтверджувати успішність змін. Захищений доступ через пароль забезпечує базовий контроль доступу без складних механізмів автентифікації.

Загалом, результати тестування підтвердили працездатність і надійність вебсистеми, її відповідність функціональним вимогам і зручність використання як для звичайних користувачів, так і для адміністратора. Система підтримує подальше масштабування, інтеграцію нових ігор і адаптацію під різні сценарії взаємодії

Крім того, була розроблена інструкція для користувачів та аміністраторів яка охоплює основні пояснення для роботи у вебзастосунку.

ВИСНОВКИ

У результаті виконання бакалаврської кваліфікаційної роботи було розроблено вебсистему для розв’язання логічних задач, яка включає такі функціональні модулі:

- модуль генерації логічних задач типу «Бики і корови» та «Задача Ейнштейна»;
- розробку режиму адміністрування, який дозволяє додавати, редагувати, видаляти логічні задачі в вебсистемі;
- розробку централізованої системи оновлення та збереження даних, яка проводить логування всіх дій користувача та адміністратора;
- розробку механізму збереження результатів, формування рейтингу користувачів та фіксації часу отримання результату.

Розробка проводилася у середовищі Visual Studio Code, яке є зручним для роботи з мовою програмування JS та PhP. Для реалізації серверної частини застосовано PhP, а для управління базами даних використано інструмент JSON-файлу. Робота оформлена з урахуванням рекомендацій [25].

У першому розділі було проведено аналіз сучасних вебсистем розв’язання логічних задач. Було виявлено їх недоліки, що стало основою для розробки власного застосунку. Другий розділ охоплював аналіз даних, розробку методу і моделі роботи системи, а також побудову алгоритмів роботи вебсторінки та його окремих модулів. У третьому розділі було обґрунтовано вибір засобів для реалізації програмного забезпечення, проведено варіантний аналіз та розроблено програмні модулі й графічний інтерфейс вебзастосунку. Четвертий розділ описує тестування розробленої вебсистеми, яке підтвердило її працездатність і відповідність очікуванням. Крім того, визначено мінімальні та рекомендовані вимоги до пристроїв для використання застосунку, а також розроблено інструкцію з використання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Третяк, Т. М. Карус як засіб розв’язування життєво важливих задач в умовах війни / Інститут психології імені Г. С. Костюка Національної академії педагогічних наук України.

2. Логіка, завдання логіки, загальне визначення, загальні відомості про логіку [Електронний ресурс] – Режим доступу: https://stud.com.ua/26847/filosofiya/logika (дата звернення: 25.03. 2025).

3. Сучасна логіка: Реферат [Електронний ресурс] – Режим доступу: https://osvita.ua/vnz/reports/logika/25243/ (дата звернення: 25.03. 2025).

4. Топольніцький, Т. В. Розробка методу і засобів реалізації вебсистеми для розв'язування логічних задач / В. В. Войтко, С. В. Бевз, Т. В. Топольніцький // Матеріали LIV Всеукраїнської науково-технічної конференції підрозділів Вінницького національного технічного університету (НТКП ВНТУ–2025) : збірник доповідей [Електронний ресурс]. 2025. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki 2025/paper/view/24104/19885 (дата звернення: 25.03.2025).

5. Білецький, Б. С. Метод оптимізації інформаційних моделей масштабованих у просторі аналітичних веб-систем за критерієм повноти їхньої топологічної спостережуваності / В. Б. Мокін, Є. М. Крижановський, А. М. Лучко, Б. С. Білецький, С. О. Жуков // Вісник Вінницького політехнічного інституту. 2021. Вип. 6. С. 131–141. DOI: https://doi.org/10.31649/1997-9266-2021-159-6-131-141.

6. Білецький, Б. С. Розроблення веб-системи обліку надання допомоги вимушено внутрішньо переміщеним особам в Україні. Інформаційно-комунікаційні технології та сталий розвиток : колективна монографія за матеріалами XXI Міжнародної науково-практичної конференції (14–16 листопада 2022 р.). Київ, 2022. С. 197–199.

7. Ключові інструменти та технології Front-end, що використовуються для розробки веб-сайтів [Електронний ресурс] – Режим

доступу: https://webbookstudio.com/ua/articles/key-tools-and-technologies-used-in-front-end-website-development/ (дата звернення: 30.03. 2025).

8. Попович, О. В. Логіка : навчальний посібник для самостійної роботи студентів. Маріуполь, 2019. 128 с.

9. Бики та корови стаття [Електронний ресурс] – Режим доступу: https://reporter.zp.ua/biki-ta-korovi-wnq.html(дата звернення: 30.03. 2025).

10. Einstein's Riddle – Step-by-step tutorial on how to solve the world's hardest puzzle [Електронний ресурс] – Режим доступу:https://medium.com/brainzilla/einsteins-riddle-step-by-step-tutorial-on-how-to-solve-the-world-s-hardest-puzzle-46bcf054a7c7
(дата звернення:30.03. 2025).

11. Генератор головоломок Zebra [Електронний ресурс] Режим доступу:
https://play.google.com/store/apps/details?hl=en_US&id=com.muroju.zebrapuzzlegenerator&utm_source=chatgpt.com&pli=1(дата звернення: 01.04. 2025).

12. Lastkatka Bot [Електронний ресурс]– Режим доступу: https://github.com/Senderman/lastkatkabot(дата звернення: 01.04. 2025).

13. Puzzling Stack Exchange [Електронний ресурс] – Режим доступу: https://puzzling.stackexchange.com (дата звернення: 01.04. 2025).

14. Freeman, A. Pro ASP. NET Core 6: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages. Berkeley, CA: Apress L. P, 2022. 1 с. ISBN 978-1-4842-7956-4.

15. JavaScript [Електронний ресурс] Режим доступу до електроного ресурса: https://developer.mozilla.org/en-US/docs/Web/JavaScript
(дата звернення: 10.04. 2025).

16. PHP [Електронний ресурс] Режим доступу: https://www.php.net/manual/en/introduction.php (дата звернення: 10.04. 2025).

17. Що таке діаграма класів [Електронний ресурс] – Режим доступу до ресурсу: https://www.mindonmap.com/uk/blog/what-is-uml-class-diagram/
(дата звернення: 11.04. 2025).

18. Лекція: Залежності між таблицями у базі даних. *JavaRush*. URL: https://javarush.com/ua/quests/lectures/ua.questhibernate.level17.lecture03 (дата звернення: 11.04.2025).

19. Comprehensive Guide to UML Activity Diagrams with Examples [Електронний ресурс] / ArchiMetric. *ArchiMetric*. 2025. URL: https://www.archimetric.com/comprehensive-guide-to-uml-activity-diagrams-with-examples/ (дата звернення: 11.04.2025).

20. What is a Flowchart? [Електронний ресурс] / Lucidchart. *Lucidchart*. URL: https://www.lucidchart.com/pages/what-is-a-flowchart (дата звернення: 11.04.2025).

21. Вступ у JSON [Електронний ресурс] Режим доступу: https://www.json.org/json-uk.html (дата звернення: 12.04. 2025).

22. MySQL Tutorial [Електронний ресурс] Режим доступу: https://www.mysqltutorial.org/(дата звернення: 12.04. 2025).

23. Що таке DOM? [Електронний ресурс] Режим доступу: https://tseivo.com/b/memecode/t/lrpjlozzyn(дата звернення: 15.04. 2025).

24. Visual Studio Code - Code Editing. Redefined Guide [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/(дата звернення: 12.04. 2025).

25. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів студентів спеціальності 121 "Інженерія програмного забезпечення" / Уклад. О. Н. Романюк, Г. О. Черноволик – Вінниця: ВНТУ, 2022. – 52с.

ДОДАТКИ

84

Додаток А. Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
«_25_» березня 2025 р.

Технічне завдання
на бакалаврську кваліфікаційну роботу
«Розробка методу та програмних засобів реалізації вебсистеми для розв'язування логічних задач»
за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської кваліфікаційної роботи:
к.т.н., доц. Войтко В. В.
«_24_» березня 2025 р.

Виконав:
студент гр. 4ПІ-21б Топольніцький Т. В.
«_24_» березня 2025 р.

Вінниця – 2025 року

1. Найменування та галузь застосування

Бакалаврська кваліфікаційна робота: «Розробка методу та програмних засобів реалізації вебсистеми для розв’язування логічних задач».

Галузь застосування – сфера самостійного навчання та інтелектуального розвитку.

2. Підстава для розробки.

Підставою для виконання бакалаврської кваліфікаційної роботи (БКР) є індивідуальне завдання на БКР та наказ №97 від 20 березня 2025 року ректора по ВНТУ про закріплення тем БКР.

3. Мета та призначення розробки.

Метою є покращення для користувачів режиму тренування логічного мислення шляхом розробки і використання спеціалізованої вебсистеми, яка забезпечує реалізацію механізму формування і перевірки задач визначених типів, фіксацію часу розв’язання задач, реалізує формуваня рейтингу користувачів і забезпечує можливість доступу з різних типів пристроїв, що дозволяє реалізувати зручний вебпортал для тренування і розвитку логічного мислення користувача.

Призначення роботи – розробка вебзастосунку для розв’язання логічних задач.

4. Вихідні дані для проведення НДР

1. Логіка, завдання логіки, загальне визначення, загальні відомості про логіку [Електронний ресурс] – Режим доступу: https://stud.com.ua/26847/filosofiya/logika (дата звернення: 25.03. 2025).

2. Ключові інструменти та технології Front-end, що використовуються для розробки веб-сайтів [Електронний ресурс] – Режим доступу: https://webbookstudio.com/ua/articles/key-tools-and-technologies-used-in-front-end-website-development/ (дата звернення: 30.03. 2025).

3. Попович, О. В. Логіка : навчальний посібник для самостійної роботи студентів. Маріуполь, 2019. 128 с.

4. Freeman, A. Pro ASP. NET Core 6: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages. Berkeley, CA: Apress L. P, 2022. 1 с. ISBN 978-1-4842-7956-4.

5. JavaScript [Електронний ресурс] Режим доступу до електроного ресурса: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 10.04. 2025).

6. Вступ у JSON [Електронний ресурс] Режим доступу: https://www.json.org/json-uk.html (дата звернення: 12.04. 2025).

5. Технічні вимоги

Комп’ютер з 64-розрядним (x64) процесором та тактовою частотою 2 ГГц. Об’єм оперативної пам’яті 2 ГБ. ПК з будь-якою операційною системою та Google Chrome 70+, Mozilla Firefox 65+, Microsoft Edge 80+, Safari 12+. Роздільна здатність екрана 1024×768. Підключення до мережі 512 Кбіт/с.

6. Конструктивні вимоги

Вебзастосунок має відповідати ергономічним та технічним вимогам. Текстова та графічна документація повинна відповідати стандартам України.

7. Перелік технічної документації, що пред’являється по закінченню робіт:

- пояснювальна записка до бакалаврської кваліфікаційної роботи;
- технічне завдання;
- лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз розвитку систем розв’язування логічних задач та постановка задачі	25.03.25 – 01.04.2025
2	Розробка методу, моделі та алгоритму роботи вебсистеми	01.04.25 – 01.05.2025
3	Розробка програмних модулів вебсистеми для розв’язування логічних задач	01.05.25 – 10.05.2025
4	Тестування програми	10.05.2025– 20.05.2025
5	Оформлення матеріалів до захисту БКР	20.05.2025 – 30.05.25

10. Порядок контролю та прийняття

Всі етапи бакалаврської кваліфікаційної роботи контролюються науковим керівником згідно плану по виконанню роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту.

88

Додаток Б. Протокол перевірки кваліфікаційної роботи

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Розробка методу та програмних засобів реалізації вебсистеми для розв'язування логічних задач»

Тип роботи: БКР

Підрозділ: кафедра програмного забезпечення, ФІТКІ

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 2,9 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- ■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту
- ☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- ☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____ _____
(прізвище, ініціали, посада) (підпис)

_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Черноволик Г. О.

З висновком експертної комісії ознайомлений

Керівник _____ Войтко В. В. к.т.н., доцент каф.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Топольніцький Т.В.
(підпис) (прізвище, ініціали)

Додаток В. Лістинг програми

```
<script>
    let bullsCows = {
        secretNumber: null,
        attempts: [],
        numberLength: 4,
        maxAttempts: 10,
        attemptsLeft: 10,
        userName: ''
    };

    let einstein = {
        attempts: [],
        maxAttempts: 10,
        attemptsLeft: 10,
        userName: '',
        grid: Array(5).fill().map(() => ({ color: '', nationality: '', drink: '', cigarette: '', pet: '' })),
        solution: null,
        clues: [],
        correctAnswer: null
    };

    const einsteinAttributes = {
        color: ['Червоний', 'Зелений', 'Білий', 'Жовтий', 'Синій'],
        nationality: ['Англієць', 'Іспанець', 'Українець', 'Норвежець', 'Японець'],
        drink: ['Кава', 'Чай', 'Молоко', 'Апельсиновий сік', 'Вода'],
        cigarette: ['Old Gold', 'Kool', 'Chesterfield', 'Lucky Strike', 'Parliament'],
```

```
    pet: ['Собака', 'Равлики', 'Лисиця', 'Кінь', 'Зебра']
};

function 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]];
    }
    return array;
}

function generateEinsteinPuzzle() {
    // Створюємо випадкове рішення
    const solution = {};
    Object.keys(einsteinAttributes).forEach(category => {
        solution[category] = shuffleArray([...einsteinAttributes[category]]);
    });

    // Вибираємо будинок із зеброю
    const zebraHouse = Math.floor(Math.random() * 5) + 1;
    einstein.correctAnswer = { house: zebraHouse, attribute: 'Зебра' };

    // Генеруємо 15 підказок
    const clues = [];
    const categories = Object.keys(einsteinAttributes);
    const usedPairs = new Set();
```

```
    // 1. Додаємо прямі підказки (наприклад, "Англієць живе в червоному будинку")
    for (let i = 0; i < 5; i++) {
        const cat1 = categories[Math.floor(Math.random() * categories.length)];
        let cat2;
        do {
            cat2 = categories[Math.floor(Math.random() * categories.length)];
        } while (cat1 === cat2 || usedPairs.has(`${cat1}-${cat2}`));
        usedPairs.add(`${cat1}-${cat2}`);
        const house = Math.floor(Math.random() * 5);
        clues.push(`${solution[cat1][house]} (з категорії ${cat1}) пов’язаний з ${solution[cat2][house]} (з категорії ${cat2}) в будинку ${house + 1}.`);
    }

    // 2. Додаємо позиційні підказки (наприклад, "Молоко п’ють у третьому будинку")
    for (let i = 0; i < 3; i++) {
        const category = categories[Math.floor(Math.random() * categories.length)];
        const house = Math.floor(Math.random() * 5);
        clues.push(`${solution[category][house]} (з категорії ${category}) знаходиться в будинку ${house + 1}.`);
    }

    // 3. Додаємо сусідські підказки (наприклад, "Норвежець живе поруч із синім будинком")
    for (let i = 0; i < 4; i++) {
        const cat1 = categories[Math.floor(Math.random() * categories.length)];
        let cat2;
        do {
```

```
            cat2 = categories[Math.floor(Math.random() * categories.length)];
        } while (cat1 === cat2);
        const house = Math.floor(Math.random() * 4); // 0–3, щоб мати сусіда
        clues.push(`${solution[cat1][house]} (з категорії ${cat1}) живе поруч із ${solution[cat2][house + 1]} (з категорії ${cat2}).`);
    }

    // 4. Додаємо відносні підказки (наприклад, "Зелений будинок ліворуч від білого")
    for (let i = 0; i < 3; i++) {
        const category = categories[Math.floor(Math.random() * categories.length)];
        const house = Math.floor(Math.random() * 4); // 0–3, щоб мати наступний будинок
        clues.push(`${solution[category][house]} (з категорії ${category}) знаходиться ліворуч від ${solution[category][house + 1]}.`);
    }

    einstein.solution = solution;
    einstein.clues = clues;

    // Оновлюємо підказки в HTML
    const cluesList = document.getElementById('einsteinClues');
    cluesList.innerHTML = clues.map(clue => `<li>${clue}</li>`).join('');
}

function generateSecretNumber(length) {
    const digits = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9];
    const result = [];
    for (let i = 0; i < length; i++) {
```

```
        const randomIndex = Math.floor(Math.random() * digits.length);
        result.push(digits.splice(randomIndex, 1)[0]);
    }
    return result.join('');
}

function validateBullsCowsInput(input, length) {
    if (input.length !== length || !/^\d+$/.test(input)) {
        return false;
    }
    const digits = new Set(input.split(''));
    return digits.size === input.length;
}

function checkBullsCowsGuess(guess, secret) {
    let bulls = 0;
    let cows = 0;
    for (let i = 0; i < secret.length; i++) {
        if (guess[i] === secret[i]) {
            bulls++;
        } else if (secret.includes(guess[i])) {
            cows++;
        }
    }
    return { bulls, cows };
}

function updateBullsCowsHistory() {
```

```
        const historyDiv = document.getElementById('history');
        historyDiv.innerHTML = bullsCows.attempts.map(attempt =>
            `<p>${attempt.guess}: ${attempt.bulls}Б ${attempt.cows}К</p>`
        ).join('');
    }

    function updateBullsCowsAttemptsLeft() {
        const attemptsLeftDiv = document.getElementById('attemptsLeft');
        attemptsLeftDiv.innerHTML = `<p>Залишилось спроб: ${bullsCows.attemptsLeft}</p>`;
    }

    function updateBullsCowsInputPlaceholder() {
        const guessInput = document.getElementById('guessInput');
        guessInput.placeholder = `Введіть ${bullsCows.numberLength} цифр`;
        guessInput.setAttribute('maxlength', bullsCows.numberLength);
    }

    function updateEinsteinAttemptsLeft() {
        const attemptsLeftDiv = document.getElementById('einsteinAttemptsLeft');
        attemptsLeftDiv.innerHTML = `<p>Залишилось спроб: ${einstein.attemptsLeft}</p>`;
    }

    function updateEinsteinGrid() {
        const cells = document.querySelectorAll('#einsteinGrid tbody td:not(:first-child)');
        cells.forEach(cell => {
```

```
        const row = parseInt(cell.dataset.row) - 1;
        const col = cell.dataset.col;
        cell.textContent = einstein.grid[row][col];
    });
}

/**
 * Надсилає запис рейтингу на сервер
 * @param {string} userName – ім’я користувача
 * @param {string} gameType – 'bullsCows' або 'einstein'
 * @param {number} score – обчислений бал
 */
function sendRatingToServer(userName, gameType, score) {
    const payload = {
        userName: userName,
        gameType: gameType,
        score: score,
        date: new Date().toISOString()
    };
    fetch('ratings.php', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json; charset=utf-8'
        },
        body: JSON.stringify(payload)
    })
    .then(res => res.json())
    .then(data => {
```

```
            console.log('Оновлений список рейтингів:', data);
            // TODO: якщо потрібно – оновити UI (наприклад, показати топ-10)
        })
        .catch(err => console.error('Помилка збереження рейтингу:', err));
    }

    function saveBullsCowsGameResult(status) {
    // 1) обчислюємо бал за формулою: (макс. спроби / фактичні спроби) * коефіцієнт
    const attempts = bullsCows.attempts.length;
    const maxAttempts = bullsCows.maxAttempts;
    const coeffs = { 3: 1, 4: 1.5, 5: 2 };
    const coefficient = coeffs[bullsCows.numberLength] || 1;
    const score = (maxAttempts / attempts) * coefficient;

    // 2) надсилаємо на сервер
    sendRatingToServer(bullsCows.userName, 'bullsCows', score);
    }

    function saveEinsteinGameResult(status) {
    // Обчислюємо бал: (макс. спроби / фактичні спроби) * 2
    const attempts = einstein.attempts.length;
    const maxAttempts = einstein.maxAttempts;
    const score = (maxAttempts / attempts) * 2;

    sendRatingToServer(einstein.userName, 'einstein', score);
    }
```

```
function displayBullsCowsHistory() {
fetch('ratings.php')
.then(res => res.json())
.then(ratings => {
   const games = ratings.filter(r => r.gameType === 'bullsCows');
   const div = document.getElementById('resultsBullsCowsHistory');
   if (games.length === 0) {
      div.innerHTML = '<p class="text-green-800">Історія ігор порожня.</p>';
      return;
   }
   div.innerHTML = games
      .map((g,i) =>
         `<p>Гра ${i+1} (${new Date(g.date).toLocaleString('uk-UA')}, ${g.userName}): бал ${g.score.toFixed(2)}</p>`
      ).join('');
})
.catch(err => console.error(err));
}



function displayEinsteinHistory() {
fetch('ratings.php')
.then(res => res.json())
.then(ratings => {
   const games = ratings.filter(r => r.gameType === 'einstein');
   const div = document.getElementById('resultsEinsteinHistory');
   if (!games.length) {
      div.innerHTML = '<p class="text-green-800">Історія порожня.</p>';
```

```
            return;
        }
        div.innerHTML = games
            .map((g,i) =>
                `<p>Гра ${i+1} (${new Date(g.date).toLocaleString('uk-UA')}, ${g.userName}): бал ${g.score.toFixed(2)}</p>`
            ).join('');
    });
    }

    function displayBullsCowsOverallRanking() {
    fetch('ratings.php')
    .then(res => res.json())
    .then(ratings => {
        const games = ratings.filter(r => r.gameType === 'bullsCows');
        const stats = {};
        games.forEach(g => {
            if (!stats[g.userName]) stats[g.userName] = { total:0, count:0 };
            stats[g.userName].total += g.score;
            stats[g.userName].count++;
        });
        const ranking = Object.entries(stats)
            .map(([user,s]) => ({ user, avg: s.total/s.count }))
            .sort((a,b) => b.avg - a.avg);
        const div = document.getElementById('resultsBullsCowsOverallRanking');
        if (!ranking.length) {
            div.innerHTML = '<p class="text-green-800">Рейтинг порожній.</p>';
            return;
```

```
    }
    div.innerHTML = ranking
      .map((r,i) =>
        `<p>${i+1}. ${r.user}: середній бал ${r.avg.toFixed(2)}</p>`
      ).join('');
  });
  }



  function displayBullsCowsSingleGameRanking() {
  fetch('ratings.php')
  .then(res => res.json())
  .then(ratings => {
    const games = ratings.filter(r => r.gameType === 'bullsCows');
    // Для кожного користувача шукаємо найвищий бал
    const best = {};
    games.forEach(g => {
      if (!best[g.userName] || g.score > best[g.userName]) {
        best[g.userName] = g.score;
      }
    });
    const ranking = Object.entries(best)
      .map(([user,score]) => ({ user, score }))
      .sort((a,b) => b.score - a.score);
    const div =
document.getElementById('resultsBullsCowsSingleGameRanking');
    if (!ranking.length) {
      div.innerHTML = '<p class="text-green-800">Рейтинг порожній.</p>';
      return;
    }
    div.innerHTML = ranking
      .map((r,i) =>
        `<p>${i+1}. ${r.user}: бал ${r.score.toFixed(2)}</p>`
      ).join('');
  })
  .catch(err => console.error('Помилка завантаження рейтингу:', err));
  }
```

```
function displayEinsteinOverallRanking() {
fetch('ratings.php')
.then(res => res.json())
.then(ratings => {
   const games = ratings.filter(r => r.gameType === 'einstein');
   // Групуємо по користувачах
   const stats = {};
   games.forEach(g => {
      if (!stats[g.userName]) stats[g.userName] = { total:0, count:0 };
      stats[g.userName].total += g.score;
      stats[g.userName].count++;
   });
   // Формуємо рейтинг за середнім балом
   const ranking = Object.entries(stats)
      .map(([user,s]) => ({ user, avg: s.total/s.count }))
      .sort((a,b) => b.avg - a.avg);
   const div = document.getElementById('resultsEinsteinOverallRanking');
   if (!ranking.length) {
      div.innerHTML = '<p class="text-green-800">Рейтинг порожній.</p>';
      return;
   }
   div.innerHTML = ranking
      .map((r,i) =>
         `<p>${i+1}. ${r.user}: середній бал ${r.avg.toFixed(2)}</p>`
      ).join('');
})
.catch(err => console.error('Помилка завантаження рейтингу:', err));
}


function displayEinsteinSingleGameRanking() {
fetch('ratings.php')
.then(res => res.json())
.then(ratings => {
   const games = ratings.filter(r => r.gameType === 'einstein');
   // Обираємо найкращий бал для кожного користувача
   const best = {};
   games.forEach(g => {
      if (!best[g.userName] || g.score > best[g.userName]) {
```

```
                best[g.userName] = g.score;
            }
        });
        const ranking = Object.entries(best)
            .map(([user,score]) => ({ user, score }))
            .sort((a,b) => b.score - a.score);
        const div =
document.getElementById('resultsEinsteinSingleGameRanking');
        if (!ranking.length) {
            div.innerHTML = '<p class="text-green-800">Рейтинг порожній.</p>';
            return;
        }
        div.innerHTML = ranking
            .map((r,i) =>
                `<p>${i+1}. ${r.user}: бал ${r.score.toFixed(2)}</p>`
            ).join('');
    })
    .catch(err => console.error('Помилка завантаження рейтингу:', err));
    }

    function endBullsCowsGame(message, status) {
        document.getElementById('result').innerHTML = message;
        document.getElementById('guessButton').disabled = true;
        document.getElementById('giveUpButton').disabled = true;
        document.getElementById('restartButton').classList.remove('hidden');
        saveBullsCowsGameResult(status);
    }

    function endEinsteinGame(message, status) {
        document.getElementById('einsteinResult').innerHTML = message;
        document.getElementById('einsteinCheckButton').disabled = true;
        document.getElementById('einsteinGiveUpButton').disabled = true;

document.getElementById('einsteinRestartButton').classList.remove('hidden');
        saveEinsteinGameResult(status);
    }

    function startNewBullsCowsGame() {
        bullsCows.numberLength =
parseInt(document.getElementById('difficulty').value);
```

```
            bullsCows.maxAttempts = bullsCows.numberLength === 3 ? 8 :
bullsCows.numberLength === 4 ? 10 : 12;
            bullsCows.attemptsLeft = bullsCows.maxAttempts;
            bullsCows.secretNumber =
generateSecretNumber(bullsCows.numberLength);
            bullsCows.attempts = [];
            document.getElementById('history').innerHTML = '';
            document.getElementById('result').innerHTML = '';
            document.getElementById('guessButton').disabled = false;
            document.getElementById('giveUpButton').disabled = false;
            document.getElementById('restartButton').classList.add('hidden');
            document.getElementById('guessInput').value = '';
            updateBullsCowsInputPlaceholder();
            updateBullsCowsAttemptsLeft();
        }

        function startNewEinsteinGame() {
            einstein.attemptsLeft = einstein.maxAttempts;
            einstein.attempts = [];
            einstein.grid = Array(5).fill().map(() => ({ color: '', nationality: '', drink: '',
cigarette: '', pet: '' }));
            generateEinsteinPuzzle();
            document.getElementById('einsteinResult').innerHTML = '';
            document.getElementById('einsteinCheckButton').disabled = false;
            document.getElementById('einsteinGiveUpButton').disabled = false;
            document.getElementById('einsteinRestartButton').classList.add('hidden');
            document.getElementById('einsteinHouse').value = '1';
            document.getElementById('einsteinAttribute').value = 'Зебра';
            updateEinsteinAttemptsLeft();
            updateEinsteinGrid();
        }

        function showMainMenu() {
            document.getElementById('mainMenu').classList.remove('hidden');
            document.getElementById('bullsCowsContainer').classList.add('hidden');
            document.getElementById('einsteinContainer').classList.add('hidden');
            document.getElementById('userNameInput').value = bullsCows.userName
|| einstein.userName;
        }

        // Обробка головного меню
```

```
        document.getElementById('startBullsCowsButton').addEventListener('click',
() => {
            const nameInput =
document.getElementById('userNameInput').value.trim();
            if (nameInput) {
                bullsCows.userName = nameInput;
                document.getElementById('mainMenu').classList.add('hidden');

document.getElementById('bullsCowsContainer').classList.remove('hidden');
                startNewBullsCowsGame();
            } else {
                alert('Будь ласка, введіть ім\'я!');
            }
        });

        document.getElementById('startEinsteinButton').addEventListener('click', ()
=> {
            const nameInput =
document.getElementById('userNameInput').value.trim();
            if (nameInput) {
                einstein.userName = nameInput;
                document.getElementById('mainMenu').classList.add('hidden');

document.getElementById('einsteinContainer').classList.remove('hidden');
                startNewEinsteinGame();
            } else {
                alert('Будь ласка, введіть ім\'я!');
            }
        });

        document.getElementById('rulesButton').addEventListener('click', () => {
            document.getElementById('rulesModal').classList.remove('hidden');
        });

        document.getElementById('closeRulesButton').addEventListener('click', () =>
{
            document.getElementById('rulesModal').classList.add('hidden');
        });
```

```
document.getElementById('showResultsButtonMenu').addEventListener('click', ()
=> {
        document.getElementById('resultsModal').classList.remove('hidden');

document.getElementById('resultsBullsCowsHistory').classList.remove('hidden');

document.getElementById('resultsBullsCowsOverallRanking').classList.add('hidde
n');

document.getElementById('resultsBullsCowsSingleGameRanking').classList.add('
hidden');
        document.getElementById('resultsEinsteinHistory').classList.add('hidden');

document.getElementById('resultsEinsteinOverallRanking').classList.add('hidden')
;

document.getElementById('resultsEinsteinSingleGameRanking').classList.add('hid
den');
        document.getElementById('showBullsCowsHistoryTab').classList.add('bg-
green-400', 'text-white');

document.getElementById('showBullsCowsHistoryTab').classList.remove('bg-
green-200', 'text-green-800');

document.getElementById('showBullsCowsOverallRankingTab').classList.add('bg
-green-200', 'text-green-800');

document.getElementById('showBullsCowsOverallRankingTab').classList.remove
('bg-green-400', 'text-white');

document.getElementById('showBullsCowsSingleGameRankingTab').classList.ad
d('bg-green-200', 'text-green-800');

document.getElementById('showBullsCowsSingleGameRankingTab').classList.re
move('bg-green-400', 'text-white');
        document.getElementById('showEinsteinHistoryTab').classList.add('bg-
green-200', 'text-green-800');

document.getElementById('showEinsteinHistoryTab').classList.remove('bg-green-
400', 'text-white');
```

```
document.getElementById('showEinsteinOverallRankingTab').classList.add('bg-green-200', 'text-green-800');

document.getElementById('showEinsteinOverallRankingTab').classList.remove('bg-green-400', 'text-white');

document.getElementById('showEinsteinSingleGameRankingTab').classList.add('bg-green-200', 'text-green-800');

document.getElementById('showEinsteinSingleGameRankingTab').classList.remove('bg-green-400', 'text-white');
        displayBullsCowsHistory();
    });

    // Обробка гри "Бики та Корови"
    document.getElementById('difficulty').addEventListener('click', () => {
        startNewBullsCowsGame();
    });

    document.getElementById('guessButton').addEventListener('click', () => {
        const guessInput = document.getElementById('guessInput');
        const guess = guessInput.value;
        const resultDiv = document.getElementById('result');

        if (!validateBullsCowsInput(guess, bullsCows.numberLength)) {
            resultDiv.innerHTML = `<p class="text-green-900">Введіть ${bullsCows.numberLength} різних цифр!</p>`;
            return;
        }

        bullsCows.attemptsLeft--;
        const { bulls, cows } = checkBullsCowsGuess(guess, bullsCows.secretNumber);
        bullsCows.attempts.push({ guess, bulls, cows });
        updateBullsCowsHistory();
        updateBullsCowsAttemptsLeft();

        if (bulls === bullsCows.numberLength) {
```

```
            endBullsCowsGame(`<p class="text-green-500">Вітаємо! Ви вгадали
число ${bullsCows.secretNumber} за ${bullsCows.attempts.length} спроб!</p>`,
'won');
        } else if (bullsCows.attemptsLeft === 0) {
            endBullsCowsGame(`<p class="text-green-900">Гра закінчена! Ви
вичерпали спроби. Число було ${bullsCows.secretNumber}.</p>`, 'lost');
        } else {
            resultDiv.innerHTML = `<p>${bulls} Биків, ${cows} Корів</p>`;
        }

        guessInput.value = '';
    });

    document.getElementById('giveUpButton').addEventListener('click', () => {
        endBullsCowsGame(`<p class="text-green-900">Ви здалися! Секретне
число було ${bullsCows.secretNumber}.</p>`, 'gave up');
    });

    document.getElementById('restartButton').addEventListener('click', () => {
        startNewBullsCowsGame();
    });

    document.getElementById('backToMenuButton').addEventListener('click', ()
=> {
        showMainMenu();
    });

    document.getElementById('showResultsButton').addEventListener('click', ()
=> {
        document.getElementById('resultsModal').classList.remove('hidden');

document.getElementById('resultsBullsCowsHistory').classList.remove('hidden');

document.getElementById('resultsBullsCowsOverallRanking').classList.add('hidde
n');

document.getElementById('resultsBullsCowsSingleGameRanking').classList.add('
hidden');
        document.getElementById('resultsEinsteinHistory').classList.add('hidden');
```

```
document.getElementById('resultsEinsteinOverallRanking').classList.add('hidden')
;

document.getElementById('resultsEinsteinSingleGameRanking').classList.add('hid
den');
        document.getElementById('showBullsCowsHistoryTab').classList.add('bg-
green-400', 'text-white');

document.getElementById('showBullsCowsHistoryTab').classList.remove('bg-
green-200', 'text-green-800');

document.getElementById('showBullsCowsOverallRankingTab').classList.add('bg
-green-200', 'text-green-800');

document.getElementById('showBullsCowsOverallRankingTab').classList.remove
('bg-green-400', 'text-white');

document.getElementById('showBullsCowsSingleGameRankingTab').classList.ad
d('bg-green-200', 'text-green-800');

document.getElementById('showBullsCowsSingleGameRankingTab').classList.re
move('bg-green-400', 'text-white');
        document.getElementById('showEinsteinHistoryTab').classList.add('bg-
green-200', 'text-green-800');

document.getElementById('showEinsteinHistoryTab').classList.remove('bg-green-
400', 'text-white');

document.getElementById('showEinsteinOverallRankingTab').classList.add('bg-
green-200', 'text-green-800');

document.getElementById('showEinsteinOverallRankingTab').classList.remove('b
g-green-400', 'text-white');

document.getElementById('showEinsteinSingleGameRankingTab').classList.add('
bg-green-200', 'text-green-800');

document.getElementById('showEinsteinSingleGameRankingTab').classList.remo
ve('bg-green-400', 'text-white');
        displayBullsCowsHistory();
```

```
    });

    // Обробка гри "Задача Ейнштейна"
    document.getElementById('einsteinGrid').addEventListener('click', (e) => {
        const cell = e.target;
        if (cell.dataset.row && cell.dataset.col) {
            const row = parseInt(cell.dataset.row) - 1;
            const col = cell.dataset.col;
            const currentValue = einstein.grid[row][col];
            const options = einsteinAttributes[col];
            const currentIndex = options.indexOf(currentValue);
            const nextIndex = (currentIndex + 1) % (options.length + 1);
            einstein.grid[row][col] = nextIndex === options.length ? '' :
options[nextIndex];
            updateEinsteinGrid();
        }
    });
    document.getElementById('einsteinCheckButton').addEventListener('click', ()
=> {
        const house = parseInt(document.getElementById('einsteinHouse').value);
        const attribute = document.getElementById('einsteinAttribute').value;
        const resultDiv = document.getElementById('einsteinResult');
        einstein.attemptsLeft--;
        einstein.attempts.push({ house, attribute });
        updateEinsteinAttemptsLeft();
        // Перевірка відповіді
        const category = Object.keys(einsteinAttributes).find(key =>
einsteinAttributes[key].includes(attribute));
        if (house === einstein.correctAnswer.house && attribute ===
einstein.correctAnswer.attribute) {
            endEinsteinGame(`<p class="text-green-500">Вітаємо! Ви правильно
визначили, що зебра в будинку ${einstein.correctAnswer.house}!</p>`, 'won');
        } else if (einstein.solution[category][house - 1] === attribute) {
            resultDiv.innerHTML = `<p class="text-green-600">Правильний
атрибут для будинку ${house}, але це не відповідає питанню про зебру.
Продовжуйте!</p>`;
        } else if (einstein.attemptsLeft === 0) {
            endEinsteinGame(`<p class="text-green-900">Гра закінчена! Ви
вичерпали спроби. Зебра була в будинку
${einstein.correctAnswer.house}.</p>`, 'lost');
        } else {
```

```
            resultDiv.innerHTML = `<p class="text-green-900">Неправильно. Спробуйте ще раз!</p>`;
        }
    });
    document.getElementById('einsteinGiveUpButton').addEventListener('click', () => {
        endEinsteinGame(`<p class="text-green-900">Ви здалися! Зебра була в будинку ${einstein.correctAnswer.house}.</p>`, 'gave up');
    });
    document.getElementById('einsteinRestartButton').addEventListener('click', () => {
        startNewEinsteinGame();
    });

document.getElementById('einsteinBackToMenuButton').addEventListener('click', () => {
        showMainMenu();
    });

document.getElementById('einsteinShowResultsButton').addEventListener('click', () => {
        document.getElementById('resultsModal').classList.remove('hidden');

document.getElementById('resultsEinsteinHistory').classList.remove('hidden');

document.getElementById('resultsBullsCowsHistory').classList.add('hidden');

document.getElementById('resultsBullsCowsOverallRanking').classList.add('hidden');

document.getElementById('resultsBullsCowsSingleGameRanking').classList.add('hidden');

document.getElementById('resultsEinsteinOverallRanking').classList.add('hidden');

document.getElementById('resultsEinsteinSingleGameRanking').classList.add('hidden');
        document.getElementById('showEinsteinHistoryTab').classList.add('bg-green-400', 'text-white');
```

```
document.getElementById('showEinsteinHistoryTab').classList.remove('bg-green-200', 'text-green-800');
        document.getElementById('showBullsCowsHistoryTab').classList.add('bg-green-200', 'text-green-800');

document.getElementById('showBullsCowsHistoryTab').classList.remove('bg-green-400', 'text-white');

document.getElementById('showBullsCowsOverallRankingTab').classList.add('bg-green-200', 'text-green-800');

document.getElementById('showBullsCowsOverallRankingTab').classList.remove('bg-green-400', 'text-white');

document.getElementById('showBullsCowsSingleGameRankingTab').classList.add('bg-green-200', 'text-green-800');

document.getElementById('showBullsCowsSingleGameRankingTab').classList.remove('bg-green-400', 'text-white');

document.getElementById('showEinsteinOverallRankingTab').classList.add('bg-green-200', 'text-green-800');

document.getElementById('showEinsteinOverallRankingTab').classList.remove('bg-green-400', 'text-white');

document.getElementById('showEinsteinSingleGameRankingTab').classList.add('bg-green-200', 'text-green-800');

document.getElementById('showEinsteinSingleGameRankingTab').classList.remove('bg-green-400', 'text-white');
        displayEinsteinHistory();
    });
    // Обробка вкладок історії та рейтингів

document.getElementById('showBullsCowsHistoryTab').addEventListener('click', () => {

document.getElementById('resultsBullsCowsHistory').classList.remove('hidden');
```

```
document.getElementById('resultsBullsCowsOverallRanking').classList.add('hidde
n');

document.getElementById('resultsBullsCowsSingleGameRanking').classList.add('
hidden');
        document.getElementById('resultsEinsteinHistory').classList.add('hidden');

document.getElementById('resultsEinsteinOverallRanking').classList.add('hidden')
;

document.getElementById('resultsEinsteinSingleGameRanking').classList.add('hid
den');
        document.getElementById('showBullsCowsHistoryTab').classList.add('bg-
green-400', 'text-white');

document.getElementById('showBullsCowsHistoryTab').classList.remove('bg-
green-200', 'text-green-800');

document.getElementById('showBullsCowsOverallRankingTab').classList.add('bg
-green-200', 'text-green-800');

document.getElementById('showBullsCowsOverallRankingTab').classList.remove
('bg-green-400', 'text-white');

document.getElementById('showBullsCowsSingleGameRankingTab').classList.ad
d('bg-green-200', 'text-green-800');

document.getElementById('showBullsCowsSingleGameRankingTab').classList.re
move('bg-green-400', 'text-white');
        document.getElementById('showEinsteinHistoryTab').classList.add('bg-
green-200', 'text-green-800');

document.getElementById('showEinsteinHistoryTab').classList.remove('bg-green-
400', 'text-white');

document.getElementById('showEinsteinOverallRankingTab').classList.add('bg-
green-200', 'text-green-800');

document.getElementById('showEinsteinOverallRankingTab').classList.remove('b
g-green-400', 'text-white');
```

```
document.getElementById('showEinsteinSingleGameRankingTab').classList.add('
bg-green-200', 'text-green-800');

document.getElementById('showEinsteinSingleGameRankingTab').classList.remo
ve('bg-green-400', 'text-white');
        displayBullsCowsHistory();
    });

document.getElementById('showBullsCowsOverallRankingTab').addEventListene
r('click', () => {

document.getElementById('resultsBullsCowsHistory').classList.add('hidden');

document.getElementById('resultsBullsCowsOverallRanking').classList.remove('h
idden');

document.getElementById('resultsBullsCowsSingleGameRanking').classList.add('
hidden');
        document.getElementById('resultsEinsteinHistory').classList.add('hidden');

document.getElementById('resultsEinsteinOverallRanking').classList.add('hidden')
;

document.getElementById('resultsEinsteinSingleGameRanking').classList.add('hid
den');
document.getElementById('showBullsCowsOverallRankingTab').classList.add('bg
-green-400', 'text-white');

document.getElementById('showBullsCowsOverallRankingTab').classList.remove
('bg-green-200', 'text-green-800');
        document.getElementById('showBullsCowsHistoryTab').classList.add('bg-
green-200', 'text-green-800');

document.getElementById('showBullsCowsHistoryTab').classList.remove('bg-
green-400', 'text-white');

document.getElementById('showBullsCowsSingleGameRankingTab').classList.ad
d('bg-green-200', 'text-green-800');
```

```
document.getElementById('showBullsCowsSingleGameRankingTab').classList.remove('bg-green-400', 'text-white');
        document.getElementById('showEinsteinHistoryTab').classList.add('bg-green-200', 'text-green-800');

document.getElementById('showEinsteinHistoryTab').classList.remove('bg-green-400', 'text-white');

document.getElementById('showEinsteinOverallRankingTab').classList.add('bg-green-200', 'text-green-800');

document.getElementById('showEinsteinOverallRankingTab').classList.remove('bg-green-400', 'text-white');

document.getElementById('showEinsteinSingleGameRankingTab').classList.add('bg-green-200', 'text-green-800');

document.getElementById('showEinsteinSingleGameRankingTab').classList.remove('bg-green-400', 'text-white');
        displayBullsCowsOverallRanking();
    });

document.getElementById('showBullsCowsSingleGameRankingTab').addEventListener('click', () => {

document.getElementById('resultsBullsCowsHistory').classList.add('hidden');

document.getElementById('resultsBullsCowsOverallRanking').classList.add('hidden');

document.getElementById('resultsBullsCowsSingleGameRanking').classList.remove('hidden');
        document.getElementById('resultsEinsteinHistory').classList.add('hidden');

document.getElementById('resultsEinsteinOverallRanking').classList.add('hidden');

document.getElementById('resultsEinsteinSingleGameRanking').classList.add('hidden');
```

```
document.getElementById('showBullsCowsSingleGameRankingTab').classList.add('bg-green-400', 'text-white');

document.getElementById('showBullsCowsSingleGameRankingTab').classList.remove('bg-green-200', 'text-green-800');
        document.getElementById('showBullsCowsHistoryTab').classList.add('bg-green-200', 'text-green-800');

document.getElementById('showBullsCowsHistoryTab').classList.remove('bg-green-400', 'text-white');

document.getElementById('showBullsCowsOverallRankingTab').classList.add('bg-green-200', 'text-green-800');

document.getElementById('showBullsCowsOverallRankingTab').classList.remove('bg-green-400', 'text-white');
        document.getElementById('showEinsteinHistoryTab').classList.add('bg-green-200', 'text-green-800');

document.getElementById('showEinsteinHistoryTab').classList.remove('bg-green-400', 'text-white');

document.getElementById('showEinsteinOverallRankingTab').classList.add('bg-green-200', 'text-green-800');

document.getElementById('showEinsteinOverallRankingTab').classList.remove('bg-green-400', 'text-white');

document.getElementById('showEinsteinSingleGameRankingTab').classList.add('bg-green-200', 'text-green-800');

document.getElementById('showEinsteinSingleGameRankingTab').classList.remove('bg-green-400', 'text-white');
        displayBullsCowsSingleGameRanking();
    });


document.getElementById('showEinsteinHistoryTab').addEventListener('click', () => {
```

```
document.getElementById('resultsEinsteinHistory').classList.remove('hidden');

document.getElementById('resultsBullsCowsHistory').classList.add('hidden');

document.getElementById('resultsBullsCowsOverallRanking').classList.add('hidde
n');

document.getElementById('resultsBullsCowsSingleGameRanking').classList.add('
hidden');

document.getElementById('resultsEinsteinOverallRanking').classList.add('hidden')
;

document.getElementById('resultsEinsteinSingleGameRanking').classList.add('hid
den');
        document.getElementById('showEinsteinHistoryTab').classList.add('bg-
green-400', 'text-white');

document.getElementById('showEinsteinHistoryTab').classList.remove('bg-green-
200', 'text-green-800');
        document.getElementById('showBullsCowsHistoryTab').classList.add('bg-
green-200', 'text-green-800');

document.getElementById('showBullsCowsHistoryTab').classList.remove('bg-
green-400', 'text-white');

document.getElementById('showBullsCowsOverallRankingTab').classList.add('bg
-green-200', 'text-green-800');

document.getElementById('showBullsCowsOverallRankingTab').classList.remove
('bg-green-400', 'text-white');

document.getElementById('showBullsCowsSingleGameRankingTab').classList.ad
d('bg-green-200', 'text-green-800');

document.getElementById('showBullsCowsSingleGameRankingTab').classList.re
move('bg-green-400', 'text-white');

document.getElementById('showEinsteinOverallRankingTab').classList.add('bg-
green-200', 'text-green-800');
```

```
document.getElementById('showEinsteinOverallRankingTab').classList.remove('bg-green-400', 'text-white');

document.getElementById('showEinsteinSingleGameRankingTab').classList.add('bg-green-200', 'text-green-800');

document.getElementById('showEinsteinSingleGameRankingTab').classList.remove('bg-green-400', 'text-white');
            displayEinsteinHistory();
        });


document.getElementById('showEinsteinOverallRankingTab').addEventListener('click', () => {
            document.getElementById('resultsEinsteinHistory').classList.add('hidden');

document.getElementById('resultsBullsCowsHistory').classList.add('hidden');

document.getElementById('resultsBullsCowsOverallRanking').classList.add('hidden');

document.getElementById('resultsBullsCowsSingleGameRanking').classList.add('hidden');

document.getElementById('resultsEinsteinOverallRanking').classList.remove('hidden');

document.getElementById('resultsEinsteinSingleGameRanking').classList.add('hidden');

document.getElementById('showEinsteinOverallRankingTab').classList.add('bg-green-400', 'text-white');

document.getElementById('showEinsteinOverallRankingTab').classList.remove('bg-green-200', 'text-green-800');
            document.getElementById('showBullsCowsHistoryTab').classList.add('bg-green-200', 'text-green-800');

document.getElementById('showBullsCowsHistoryTab').classList.remove('bg-green-400', 'text-white');
```

```
document.getElementById('showBullsCowsOverallRankingTab').classList.add('bg
-green-200', 'text-green-800');

document.getElementById('showBullsCowsOverallRankingTab').classList.remove
('bg-green-400', 'text-white');

document.getElementById('showBullsCowsSingleGameRankingTab').classList.ad
d('bg-green-200', 'text-green-800');

document.getElementById('showBullsCowsSingleGameRankingTab').classList.re
move('bg-green-400', 'text-white');
        document.getElementById('showEinsteinHistoryTab').classList.add('bg-
green-200', 'text-green-800');

document.getElementById('showEinsteinHistoryTab').classList.remove('bg-green-
400', 'text-white');

document.getElementById('showEinsteinSingleGameRankingTab').classList.add('
bg-green-200', 'text-green-800');

document.getElementById('showEinsteinSingleGameRankingTab').classList.remo
ve('bg-green-400', 'text-white');
        displayEinsteinOverallRanking();
    });

document.getElementById('showEinsteinSingleGameRankingTab').addEventListe
ner('click', () => {
        document.getElementById('resultsEinsteinHistory').classList.add('hidden');

document.getElementById('resultsBullsCowsHistory').classList.add('hidden');

document.getElementById('resultsBullsCowsOverallRanking').classList.add('hidde
n');
document.getElementById('resultsBullsCowsSingleGameRanking').classList.add('
hidden');
document.getElementById('resultsEinsteinOverallRanking').classList.add('hidden';

document.getElementById('resultsEinsteinSingleGameRanking').classList.remove(
'hidden');
```

```
document.getElementById('showEinsteinSingleGameRankingTab').classList.add('
bg-green-400', 'text-white');

document.getElementById('showEinsteinSingleGameRankingTab').classList.remo
ve('bg-green-200', 'text-green-800');
        document.getElementById('showBullsCowsHistoryTab').classList.add('bg-
green-200', 'text-green-800');

document.getElementById('showBullsCowsHistoryTab').classList.remove('bg-
green-400', 'text-white');

document.getElementById('showBullsCowsOverallRankingTab').classList.add('bg
-green-200', 'text-green-800');

document.getElementById('showBullsCowsOverallRankingTab').classList.remove
('bg-green-400', 'text-white');

document.getElementById('showBullsCowsSingleGameRankingTab').classList.ad
d('bg-green-200', 'text-green-800');

document.getElementById('showBullsCowsSingleGameRankingTab').classList.re
move('bg-green-400', 'text-white');
        document.getElementById('showEinsteinHistoryTab').classList.add('bg-
green-200', 'text-green-800');

document.getElementById('showEinsteinHistoryTab').classList.remove('bg-green-
400', 'text-white');
document.getElementById('showEinsteinOverallRankingTab').classList.add('bg-
green-200', 'text-green-800');

document.getElementById('showEinsteinOverallRankingTab').classList.remove('b
g-green-400', 'text-white');
        displayEinsteinSingleGameRanking();
    });
    document.getElementById('closeModalButton').addEventListener('click', ()
=> {
        document.getElementById('resultsModal').classList.add('hidden');
    });
</script>
```

Додаток Г. Ілюстративна частина

Рисунок Г.1 – Титульний слайд

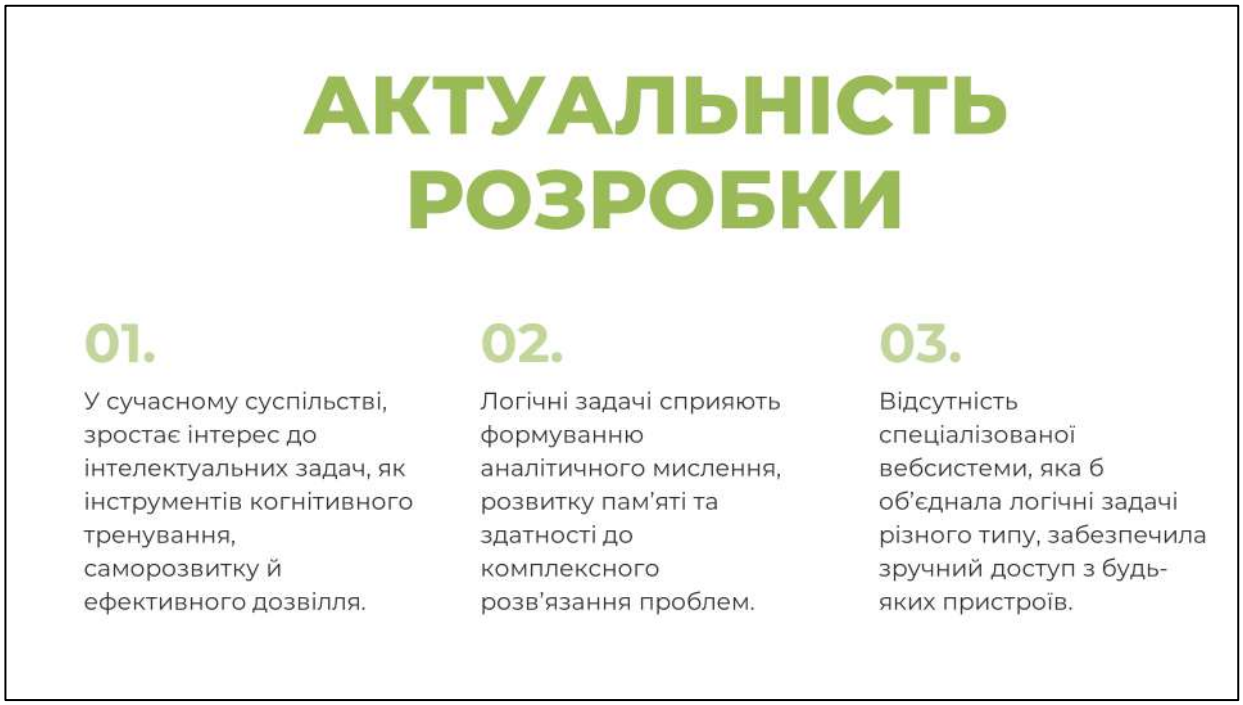


Рисунок Г.2 – Актуальність розробки

Рисунок Г.3 – Мета дослідження

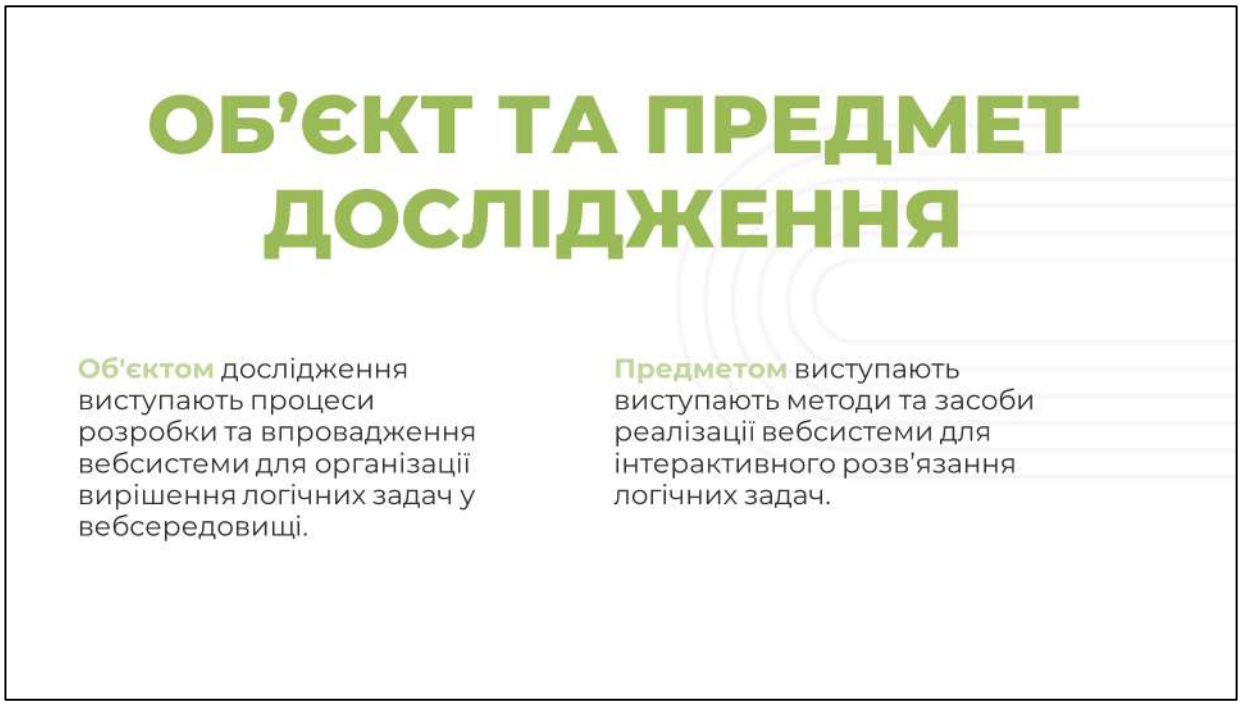


Рисунок Г.4 – Об’єкт та предмет дослідження

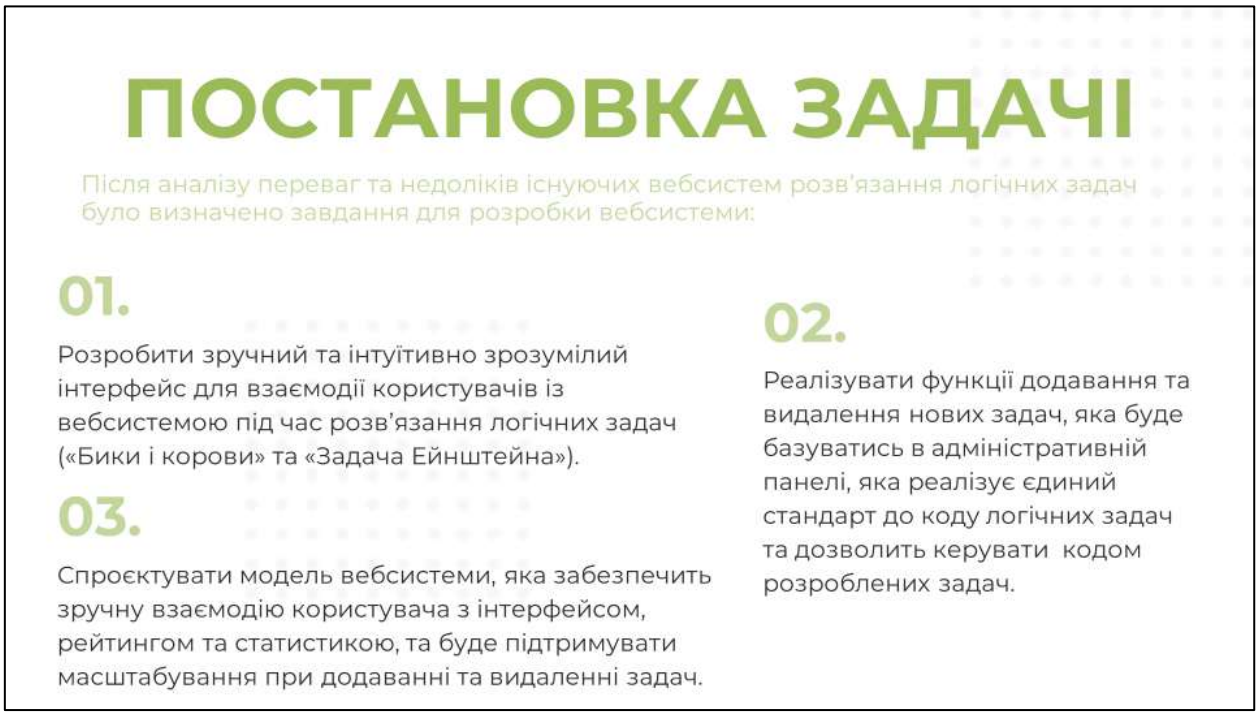


Рисунок Г.5 – Постановка задачі

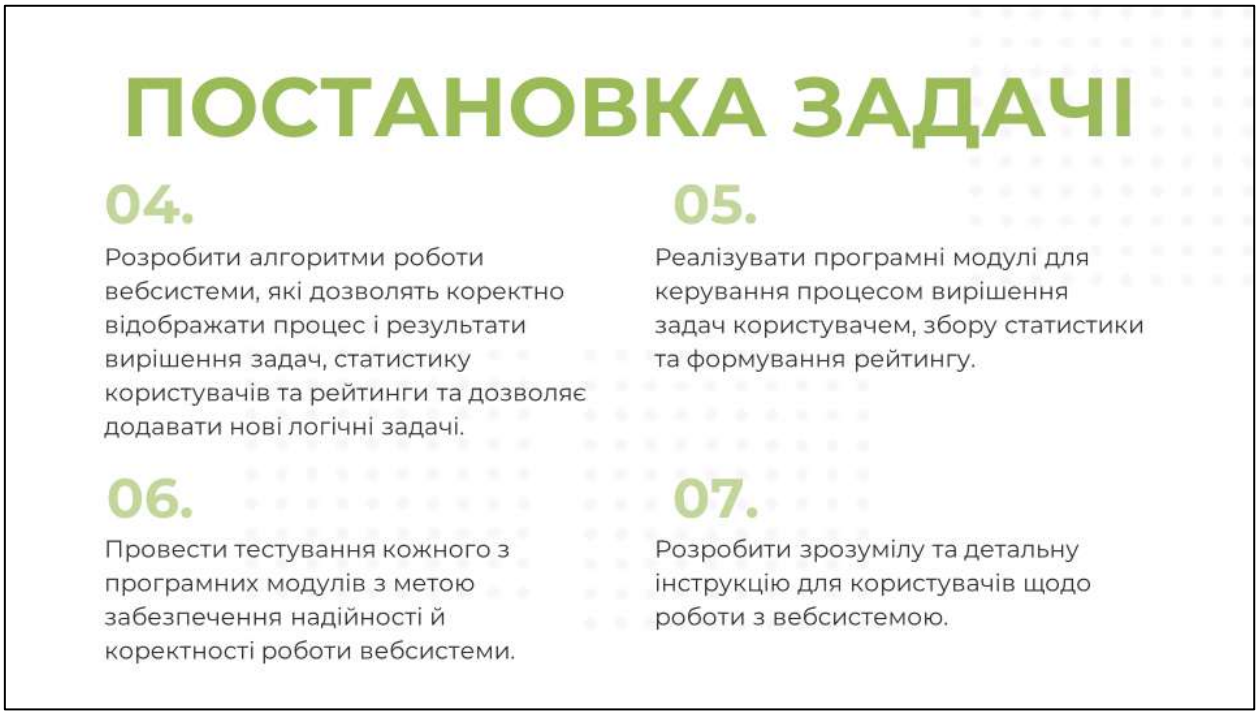


Рисунок Г.6 – Продовження постановки задачі

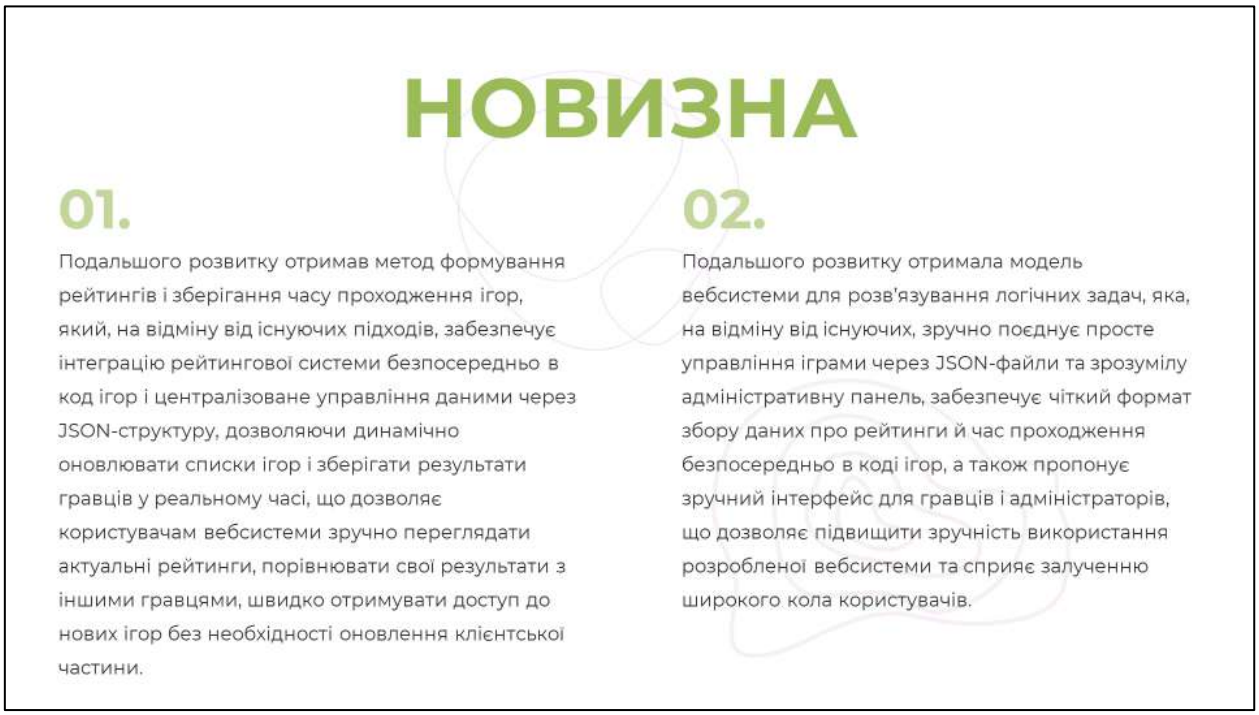


Рисунок Г.7 – Новизна розробленого застосунку

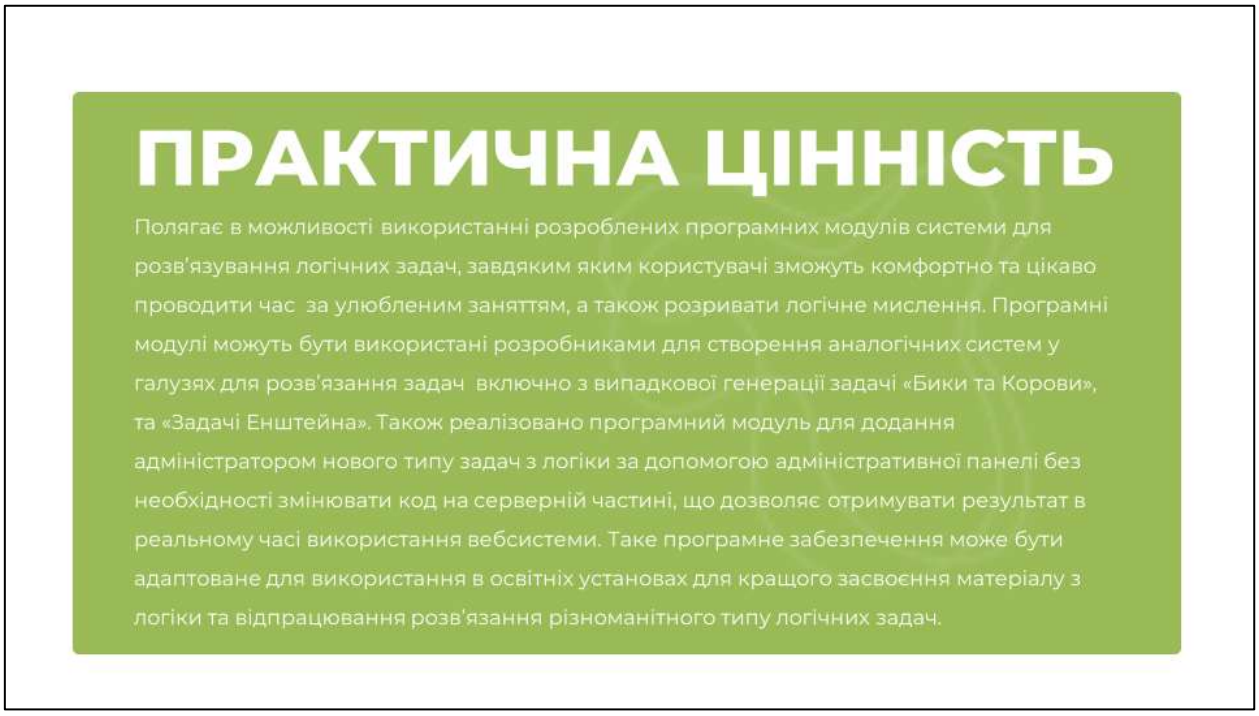


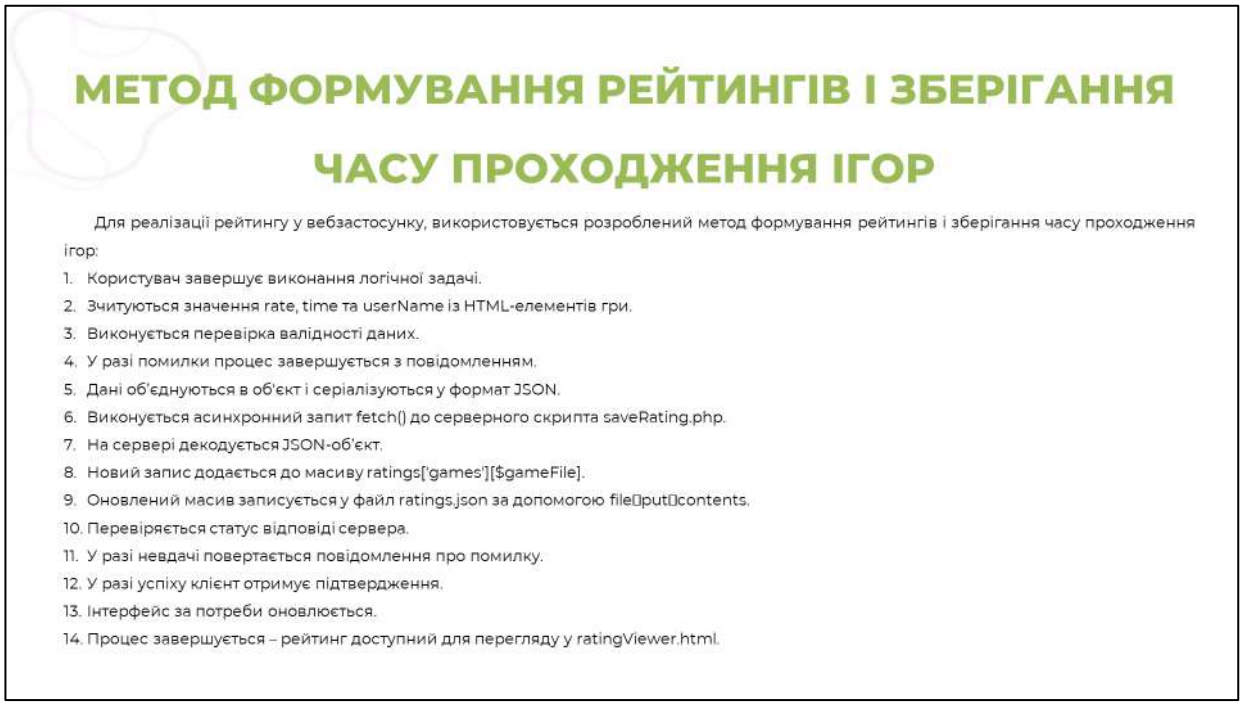
Рисунок Г.8 – Практична цінність

АНАЛІЗ АНАЛОГІВ

Критерій порівняння	Генератор головоломок Zebra	Lastkatka Bot	Puzzling Stack	Власна розробка
Можливість розв'язувати логічні задачі типу «Задачі Ейнштейна»	1	0	0	1
Можливість розв'язувати логічні задачі типу «Бики і корови»	0	1	0	1
Можливостю додання нового типу логічних задач за допомогою адміністративної консолі	0	0	0	1
Доступність на всіх девайсах без особливих вимог	0	0	1	1
Збереження результатів часу розв'язування	0	1	0	1
Можливість зв'язку зі спеціалістом у чаті в застосунку	0	1	1	1
Сумарний коефіцієнт	**1**	**3**	**2**	**6**

Результати даного етапу дослідження опубліковані в тезах доповіді та апробовані на конференції.

Рисунок Г.9 – Аналіз аналогів



МЕТОД ФОРМУВАННЯ РЕЙТИНГІВ І ЗБЕРІГАННЯ ЧАСУ ПРОХОДЖЕННЯ ІГОР

Для реалізації рейтингу у вебзастосунку, використовується розроблений метод формування рейтингів і зберігання часу проходження ігор:

1. Користувач завершує виконання логічної задачі.
2. Зчитуються значення rate, time та userName із HTML-елементів гри.
3. Виконується перевірка валідності даних.
4. У разі помилки процес завершується з повідомленням.
5. Дані об'єднуються в об'єкт і серіалізуються у формат JSON.
6. Виконується асинхронний запит fetch() до серверного скрипта saveRating.php.
7. На сервері декодується JSON-об'єкт.
8. Новий запис додається до масиву ratings['games'][$gameFile].
9. Оновлений масив записується у файл ratings.json за допомогою file□put□contents.
10. Перевіряється статус відповіді сервера.
11. У разі невдачі повертається повідомлення про помилку.
12. У разі успіху клієнт отримує підтвердження.
13. Інтерфейс за потреби оновлюється.
14. Процес завершується – рейтинг доступний для перегляду у ratingViewer.html.

Рисунок Г.10 – Метод формування рейтингів і зберігання часу проходження ігор

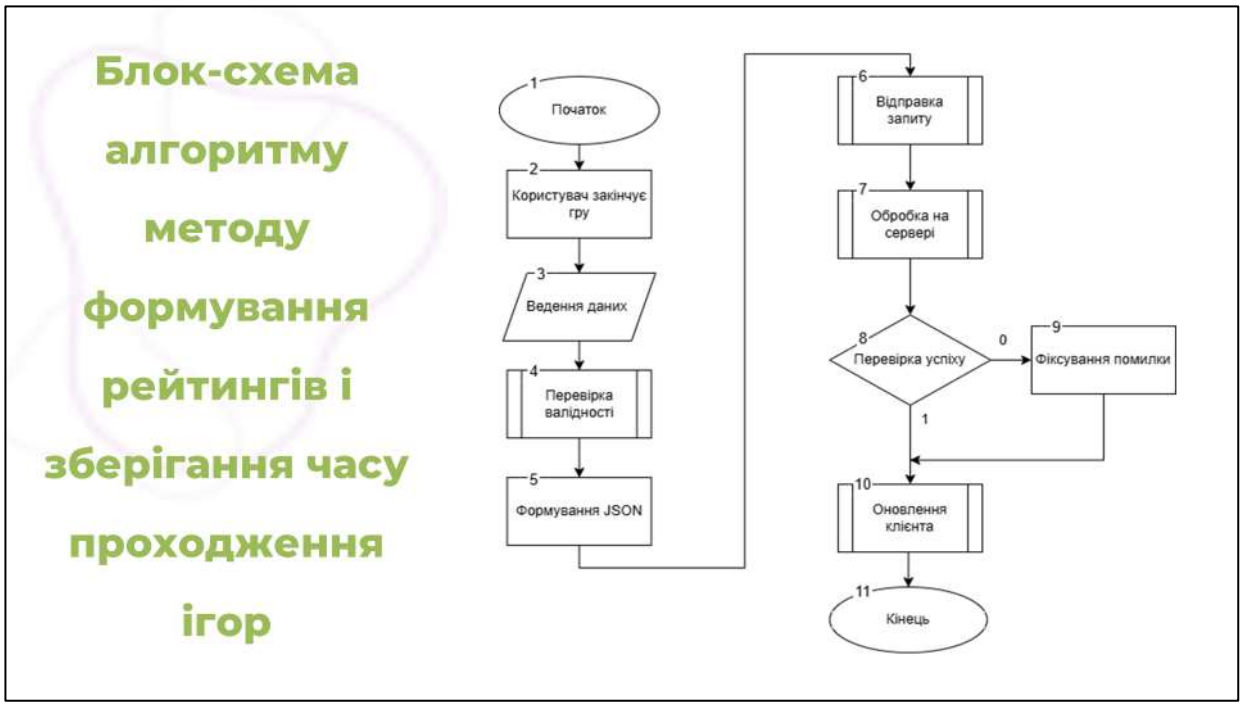


Рисунок Г.11 – Блок-схема алгоритму методу формування рейтингів і зберігання часу проходження ігор

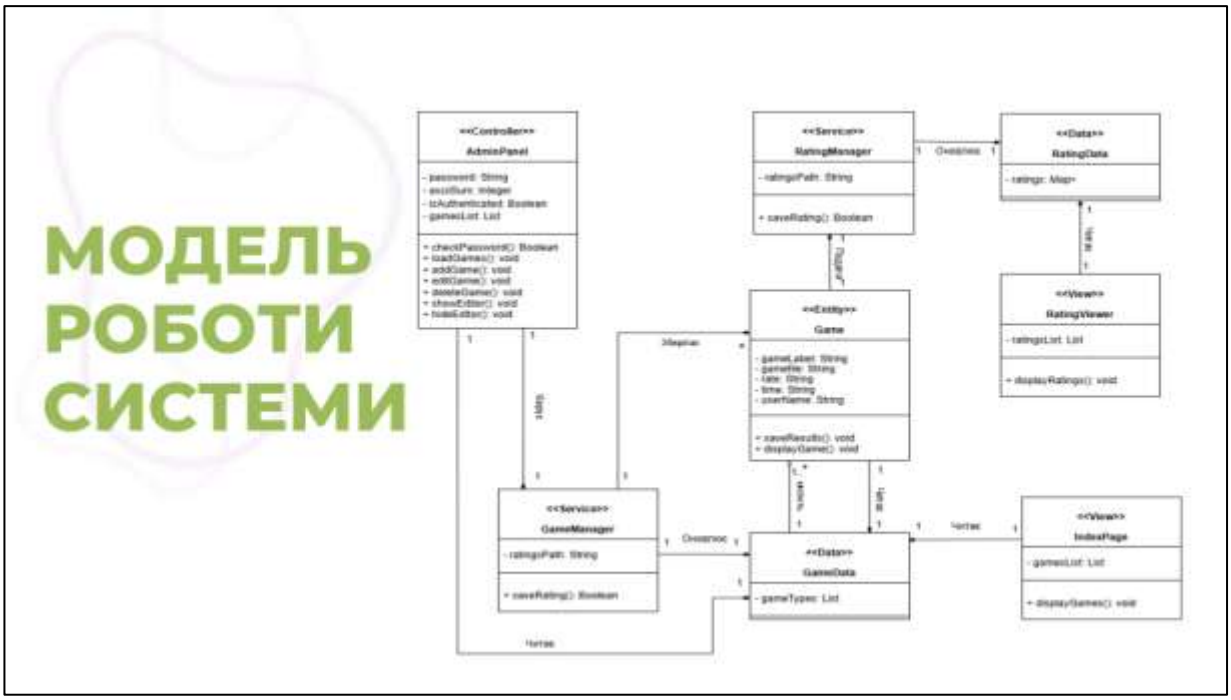


Рисунок Г.12 – Модель роботи системи

Рисунок Г.13 – Загальний алгоритм роботи системи

Рисунок Г.14 – Діаграми діяльності

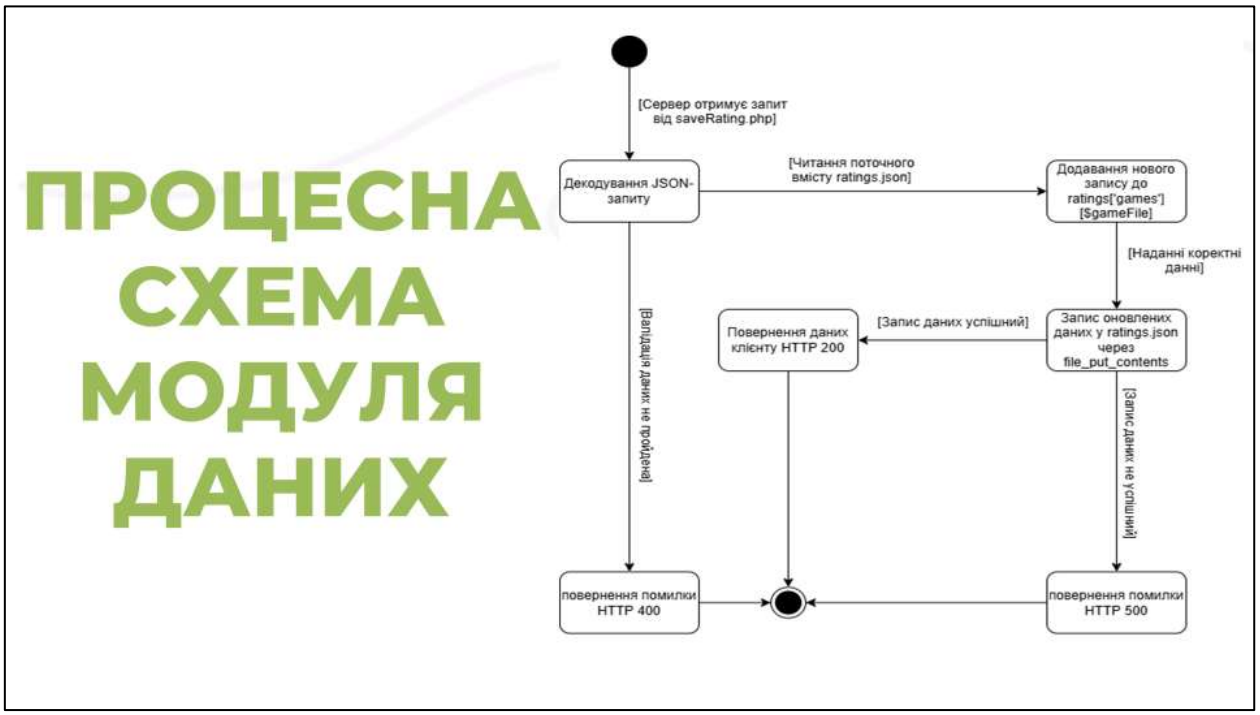


Рисунок Г.15 – Процесна схема модуля даних

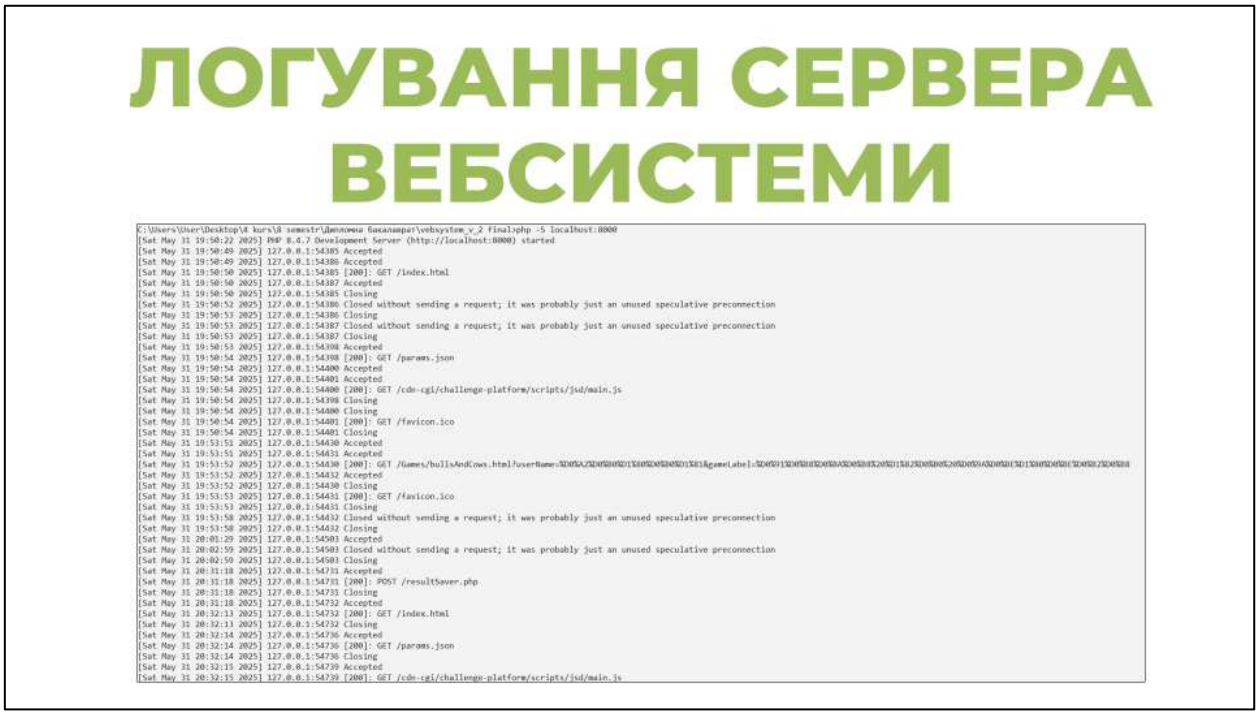


Рисунок Г.16 – Логування сервера вебсистеми

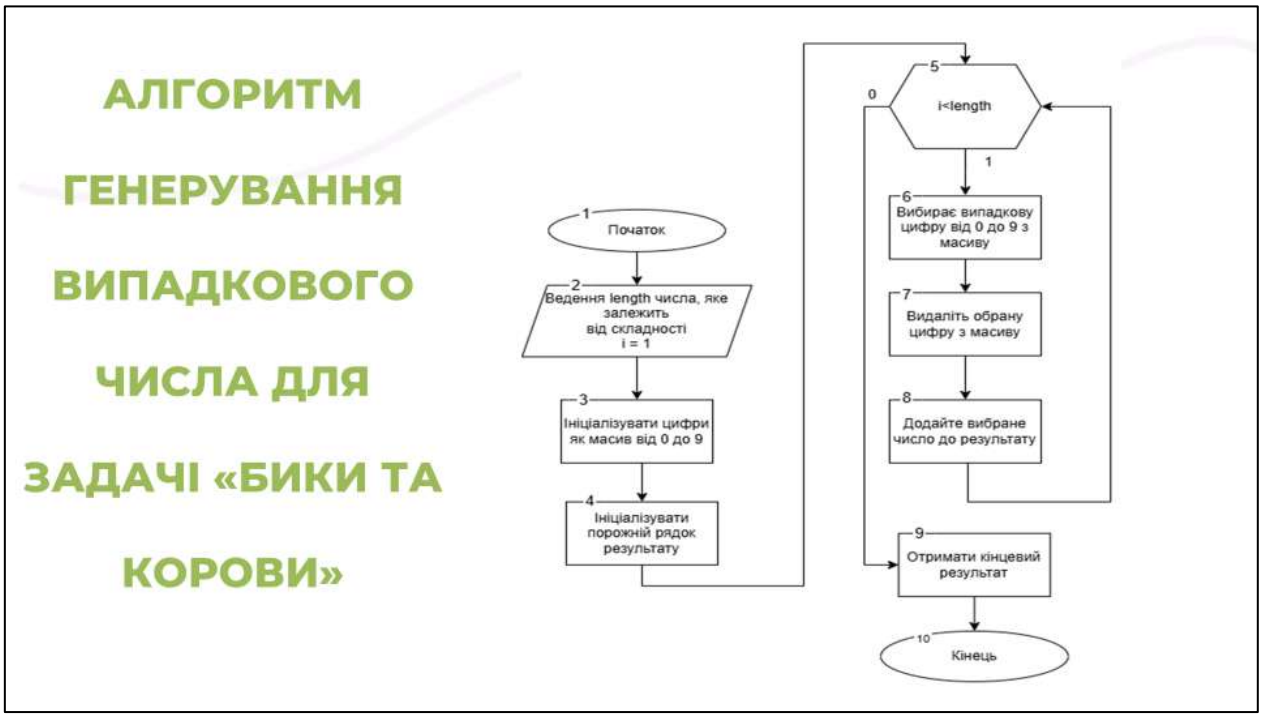


Рисунок Г.17 – Алгоритм генерування випадкового числа для задачі «Бики та корови»

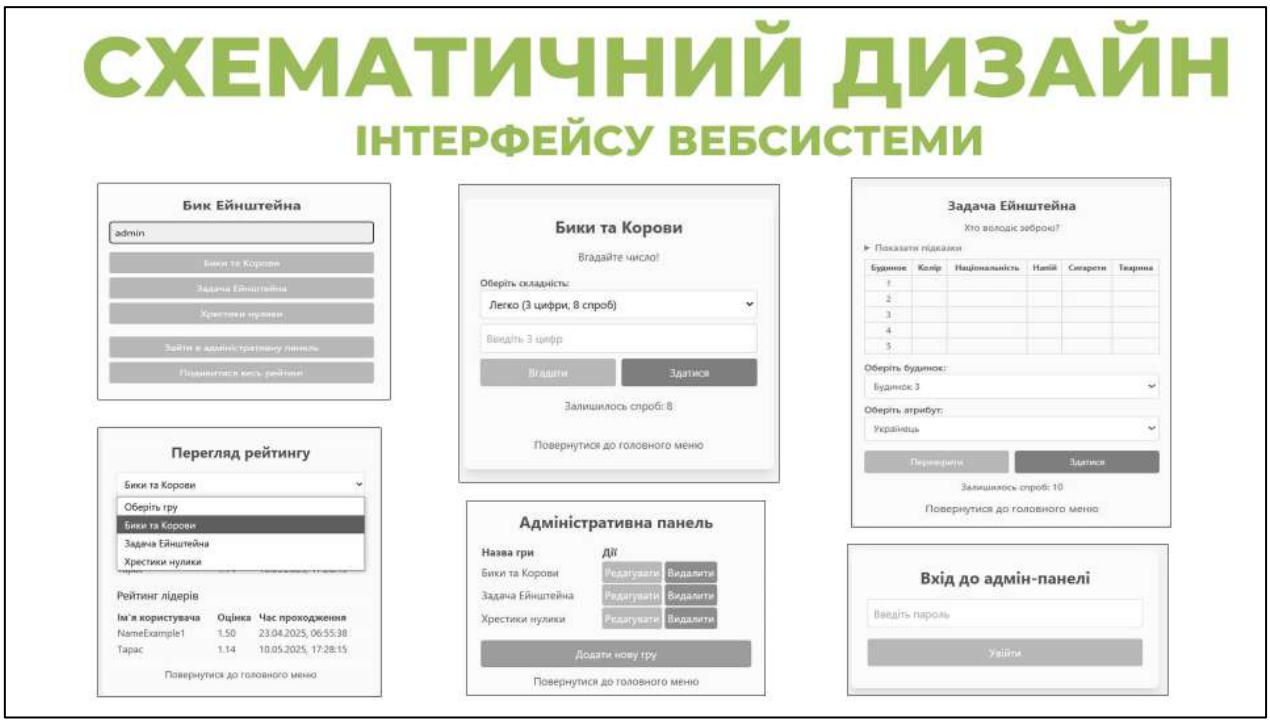


Рисунок Г.18 – Схематичний дизайн інтерфейсу вебсистеми

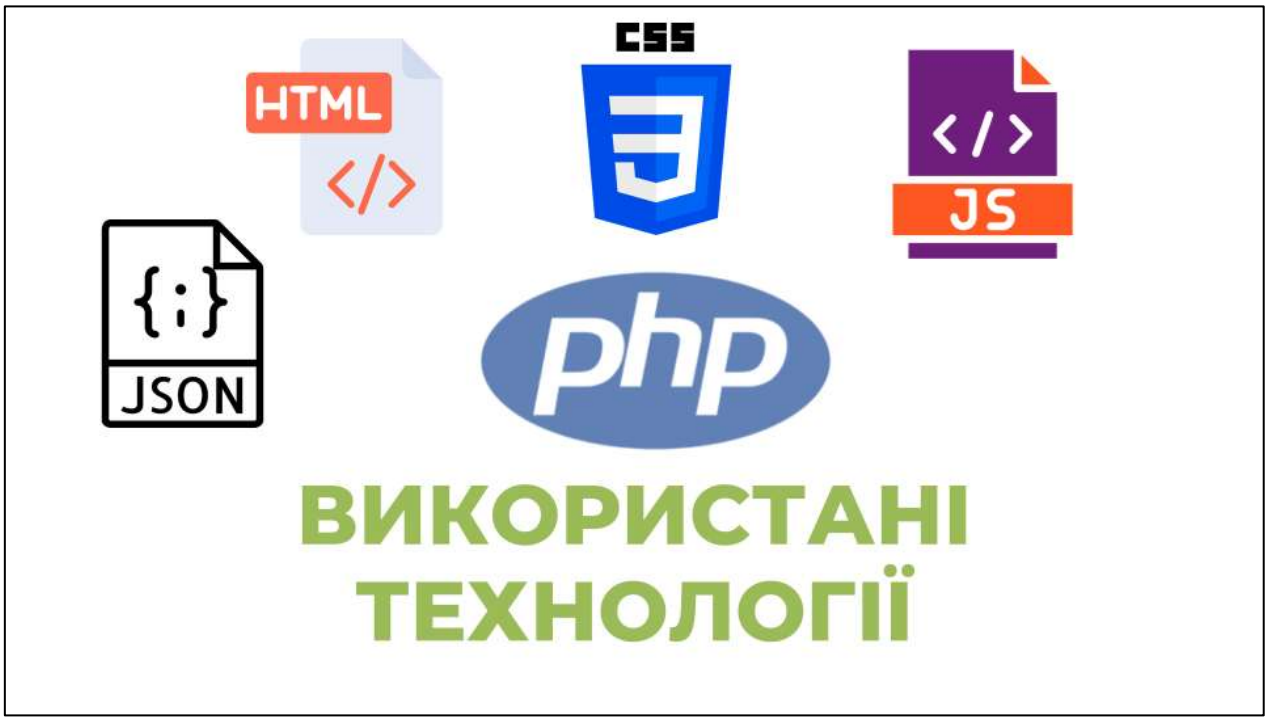


Рисунок Г.19 – Використані технології

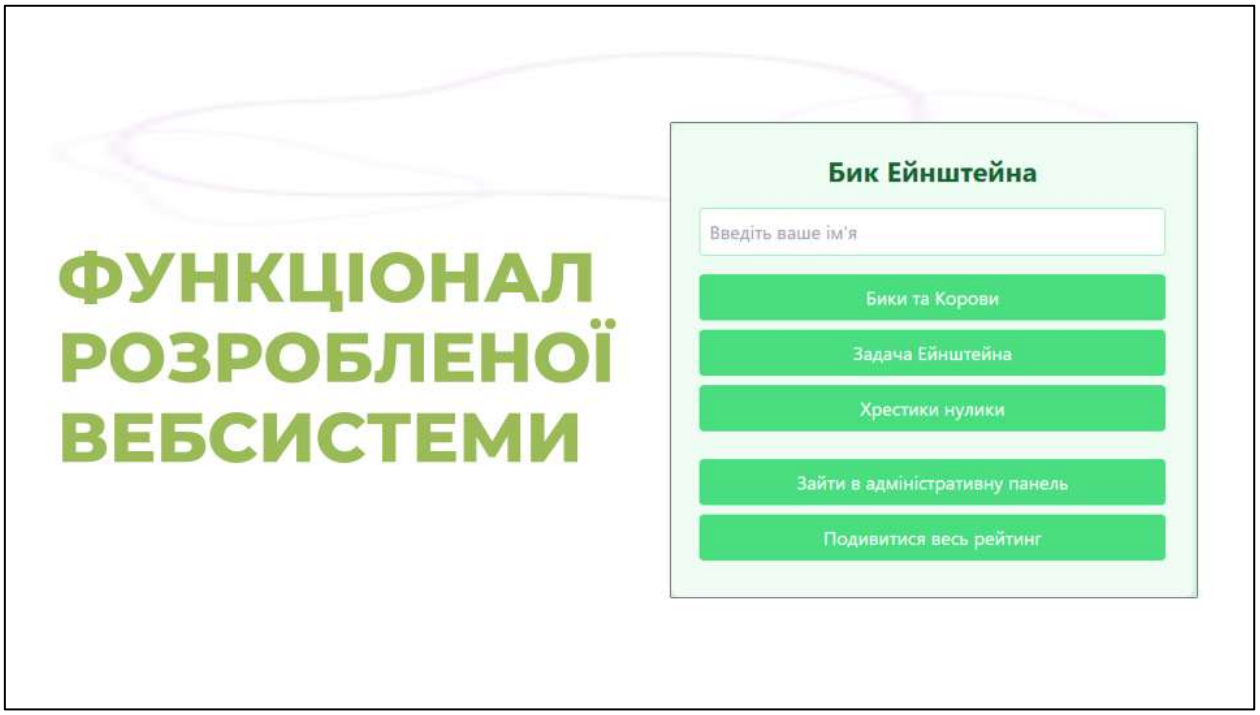


Рисунок Г.20 – Функціонал розробленої вебсистеми

Рисунок Г.21 – Функціонал розробленої вебсистеми валідація імені

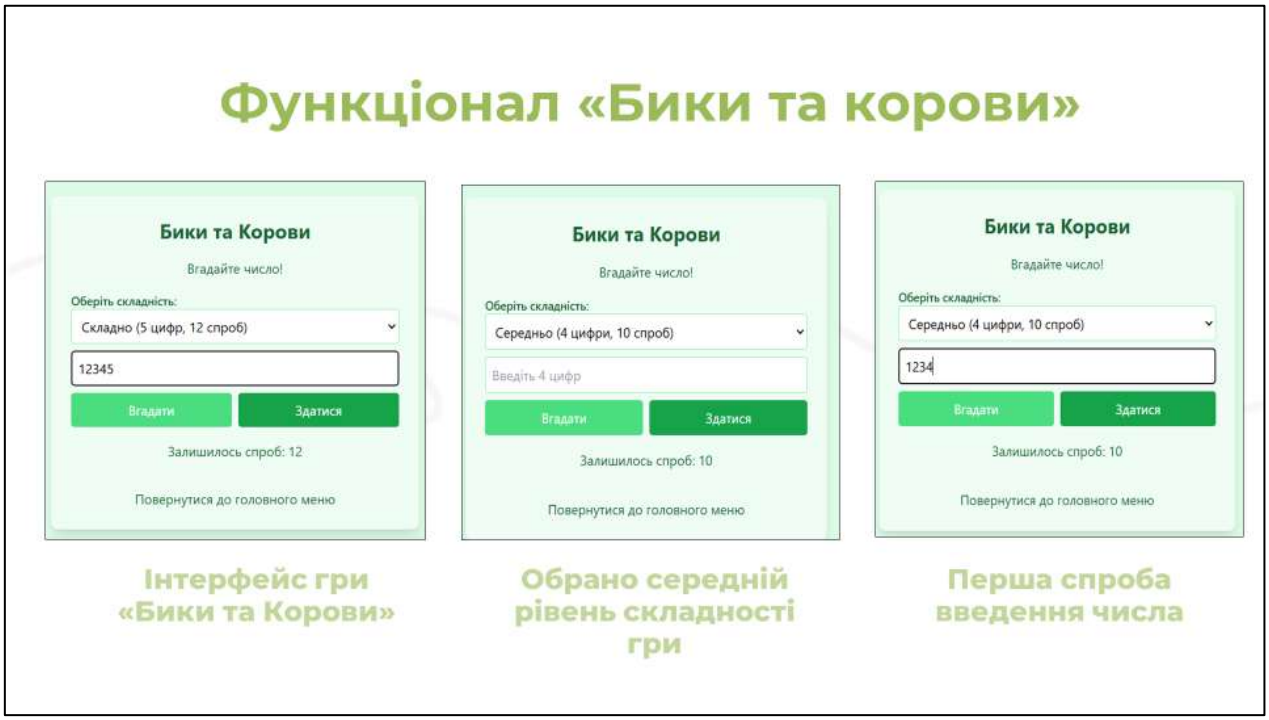


Рисунок Г.22 – Функціонал розробленої вебсистеми модуля «Бики та корови»

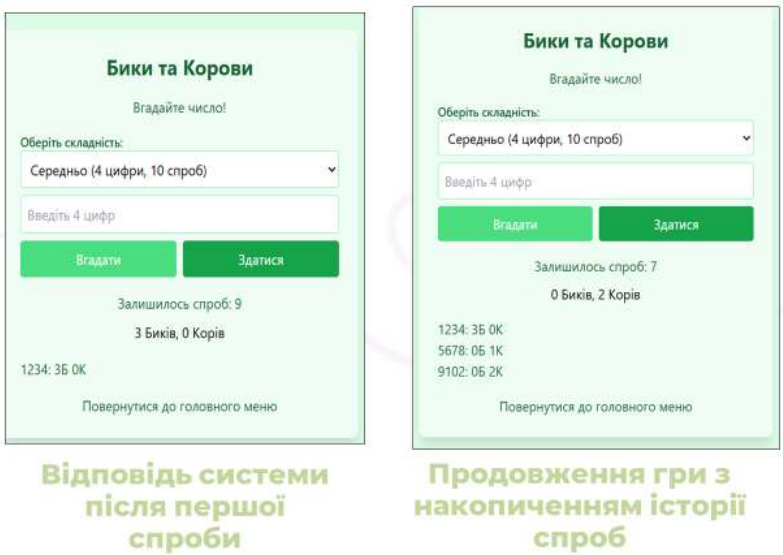


Рисунок Г.23 – Продовження функціоналу розробленої вебсистеми модуля «Бики та корови»

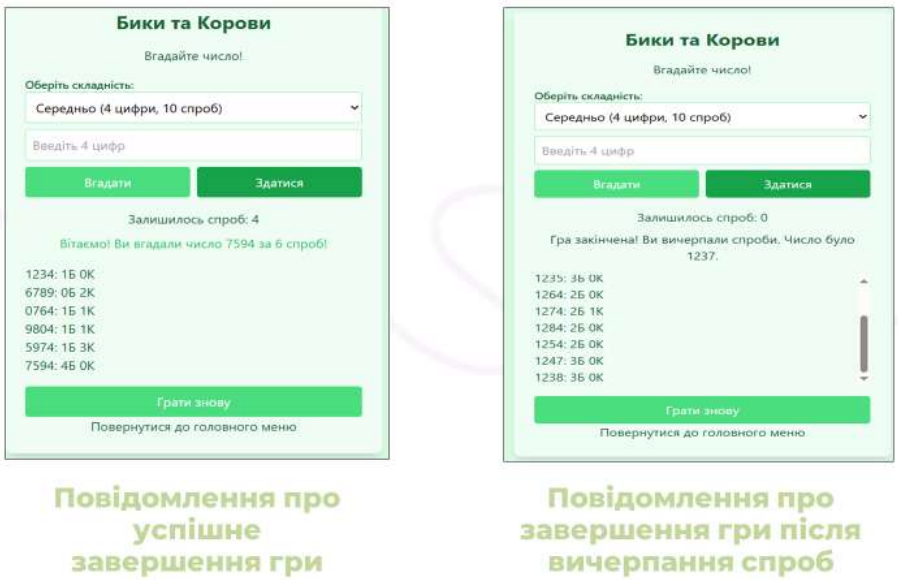


Рисунок Г.24 – Продовження функціоналу розробленої вебсистеми модуля «Бики та корови»

Рисунок Г.25 – Функціонал розробленої вебсистеми модуля «Задачі Енштейна»

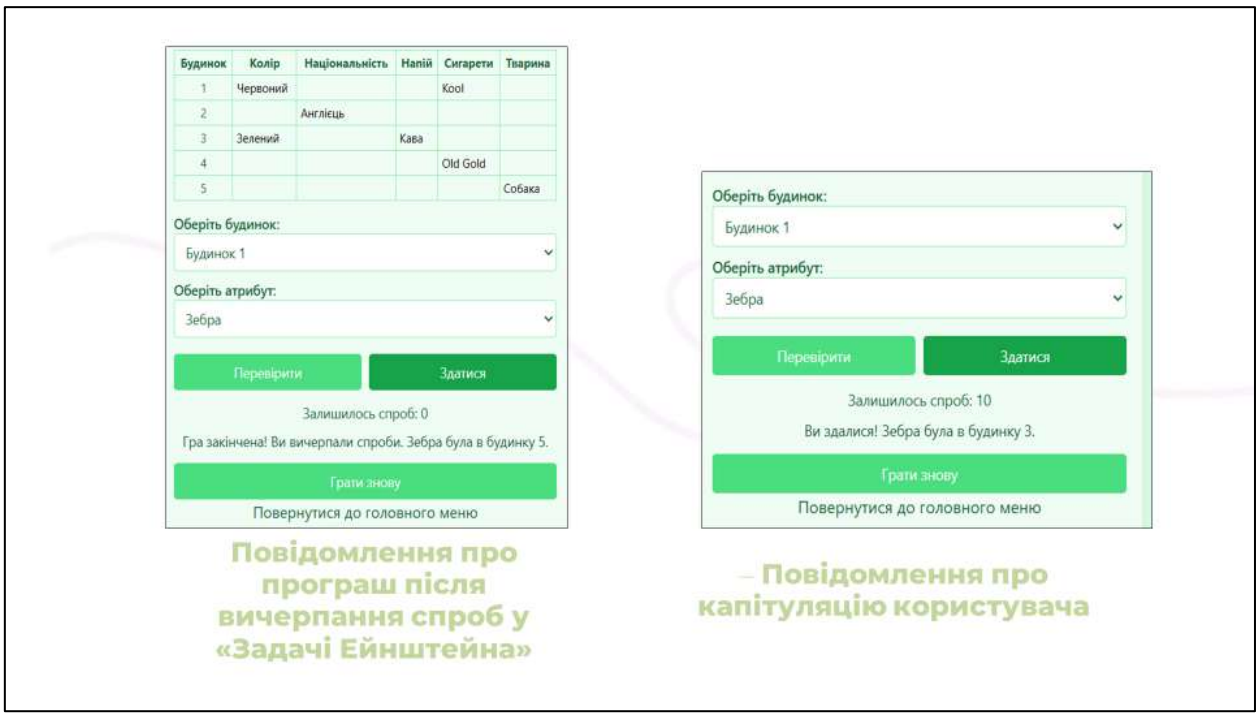


Рисунок Г.26 – Продовження функціоналу розробленої вебсистеми модуля «Задачі Енштейна»

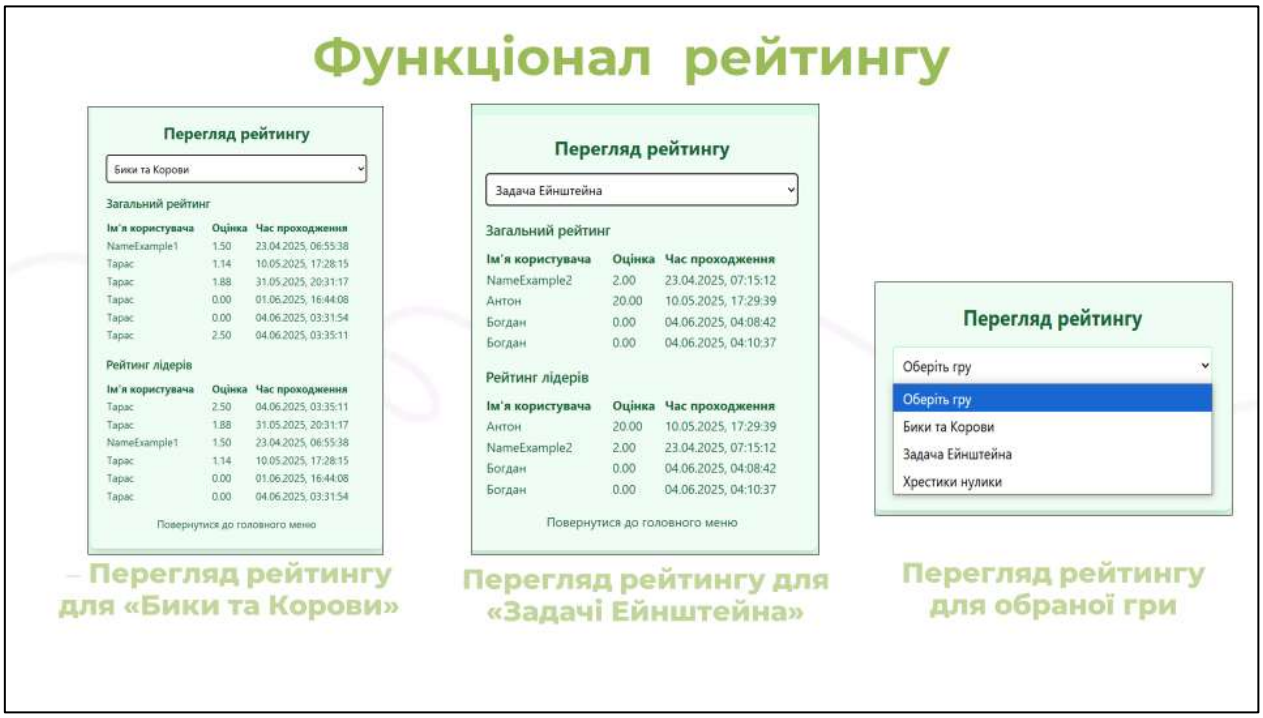


Рисунок Г.27 – Функціонал розробленої вебсистеми модуля рейтингу

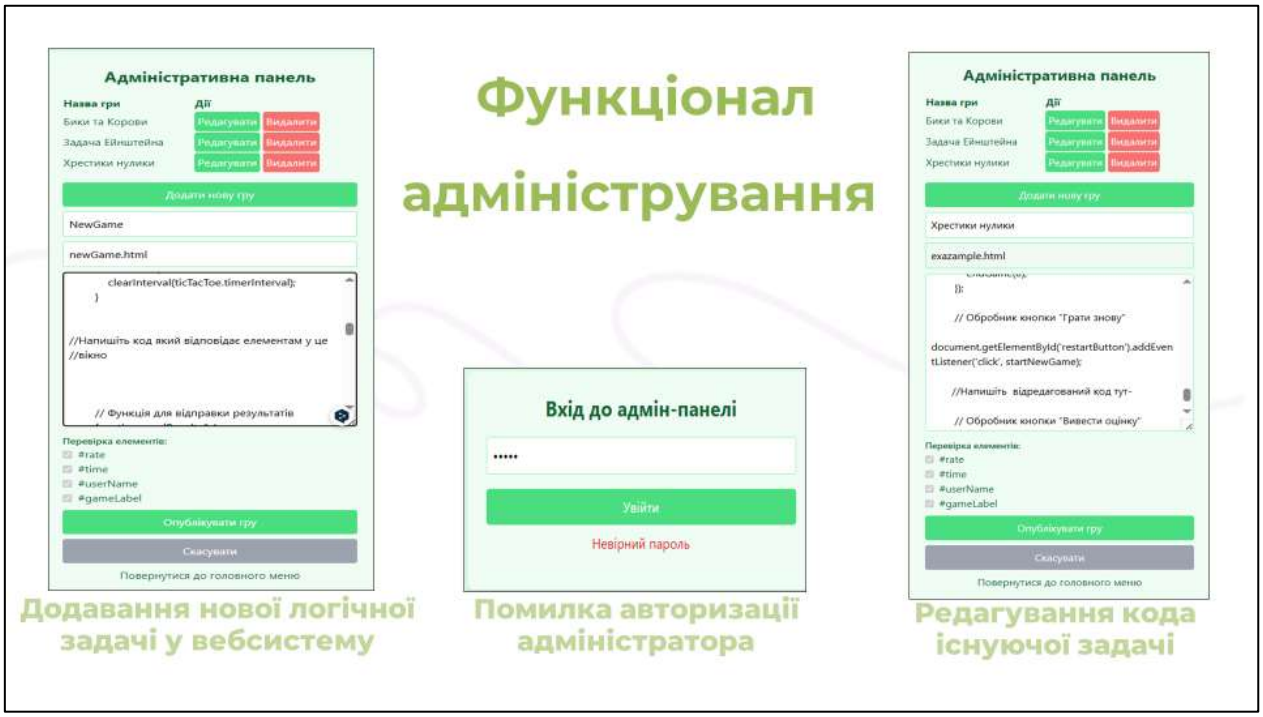


Рисунок Г.28 – Функціонал розробленої вебсистеми модуля адміністрування

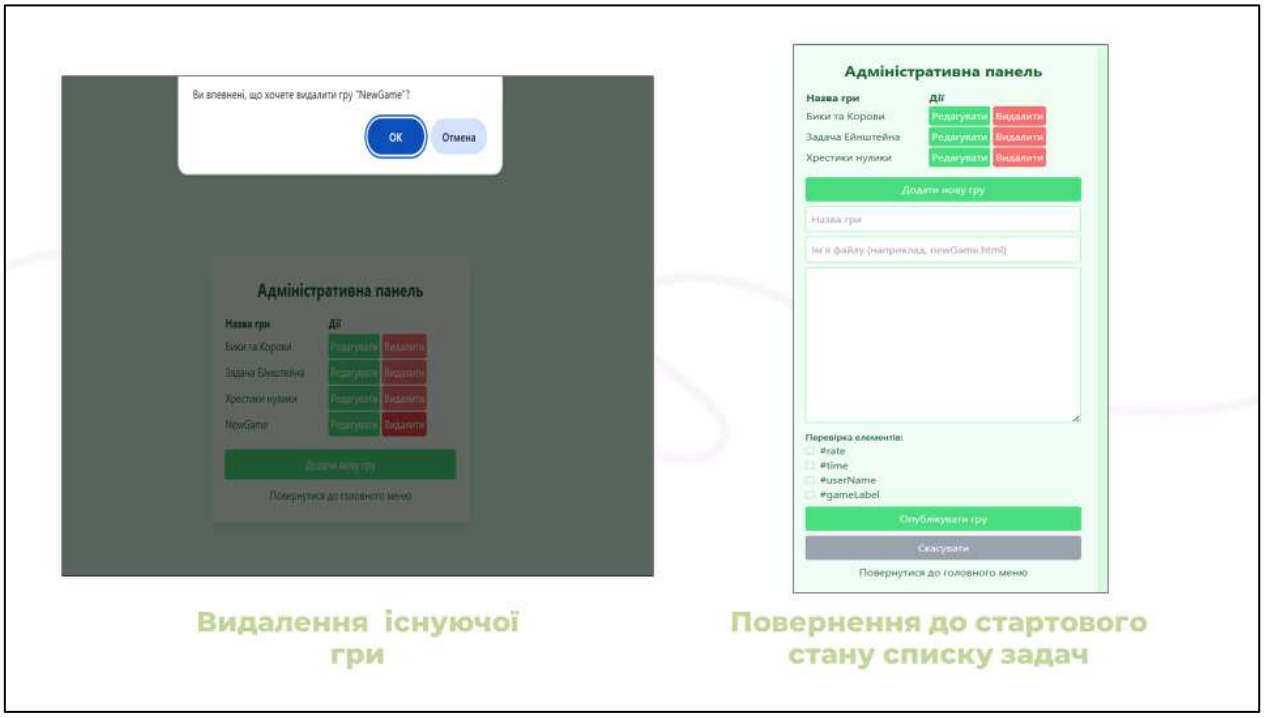


Рисунок Г.29 – Продовження функціоналу розробленої вебсистеми модуля адміністрування

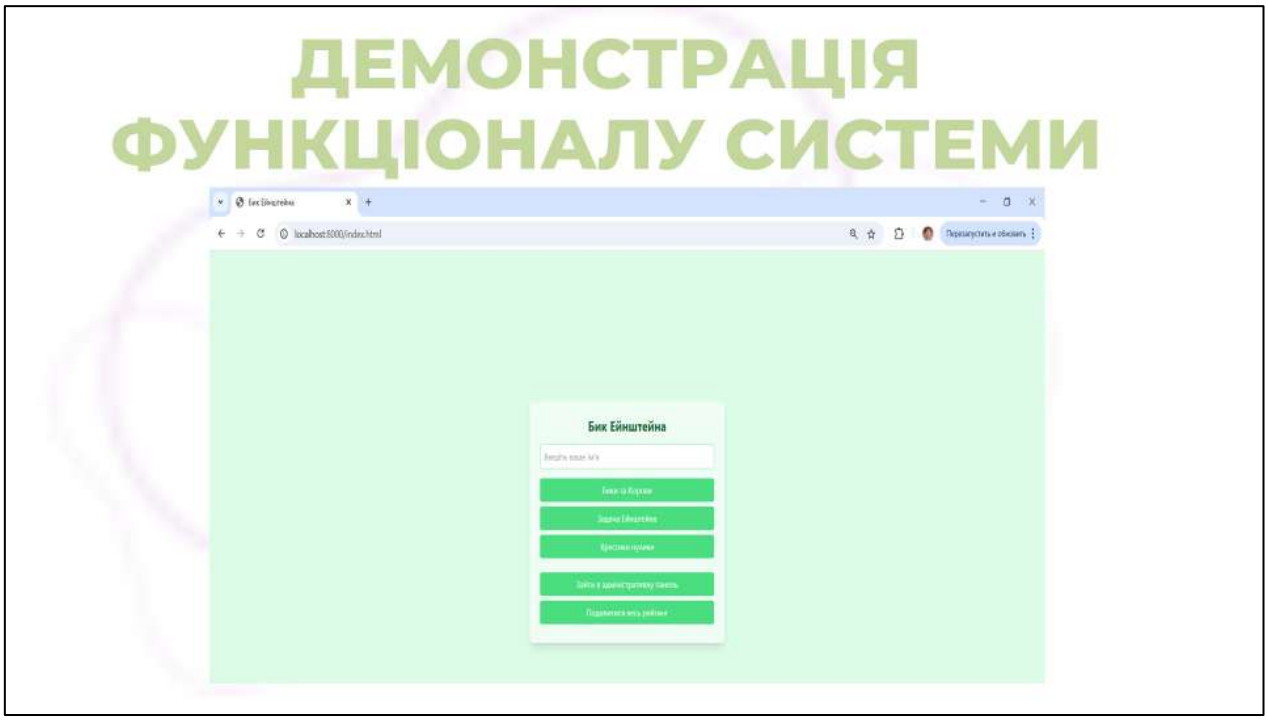


Рисунок Г.30 – Демонстрація функціоналу системи

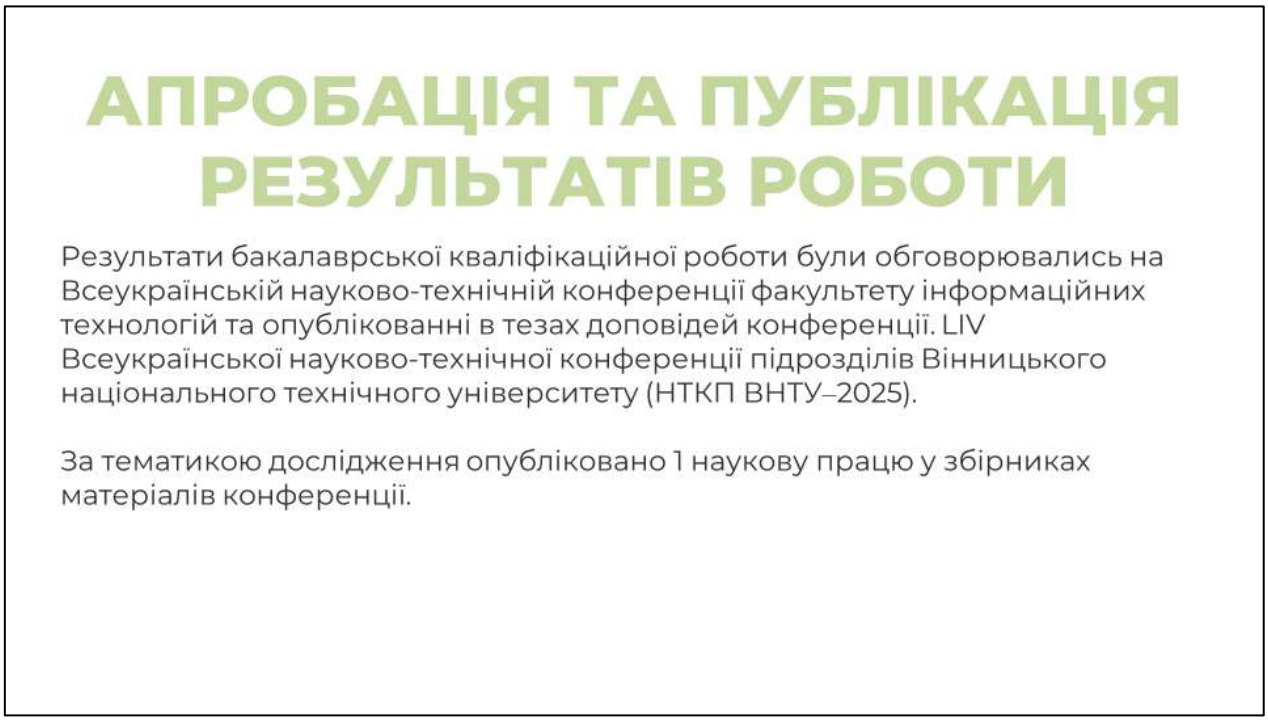


Рисунок Г.31 – Апробація та публікація результатів роботи

Рисунок Г.32 – Висновки