Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Ігровий програмний застосунок в жанрі Multiplayer Party Games. Хост частина, левел-дизайн та UI/UX.
(тема)

Виконав:
студент 4 курсу, групи ПЗПІ-20-10

Двугрошев А. О.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник ст. викл. кафедри ПІ Новіков Ю. С.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З. В. Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Двугрошев Андрій Олексійович
(прізвище, ім'я, по батькові)

1. Тема роботи Ігровий програмний застосунок в жанрі Multiplayer Party Games. Хост частина, левел-дизайн та UI/UX.
Затверджена наказом по університету від 20.05.2024р. № 471 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 06.06.2024
3. Вихідні дані до роботи методичні вказівки до кваліфікаційної роботи бакалавра за спеціальністю 121 – інженерія програмного забезпечення освітньо-професійна програма «програмна інженерія» для здобувачів усіх форм навчання.
4. Перелік питань, що потрібно опрацювати в роботі
вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проєктування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, впровадження програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	08.04.2024	*виконано*
2	Створення специфікації ПЗ	10.04.2024	*виконано*
3	Проєктування ПЗ	16.04.2024	*виконано*
4	Розробка ПЗ	16.05.2024	*виконано*
5	Тестування ПЗ	22.05.2024	*виконано*
6	Оформлення пояснювальної записки	26.05.2024	*виконано*
7	Підготовка презентації та доповіді	28.05.2024	*виконано*
8	Попередній захист	30.05.2024	*виконано*
9	Нормоконтроль, рецензування	02.06.2024	*виконано*
10	Здача роботи у електронний архів	05.06.2024	*виконано*
11	Допуск до захисту у зав. кафедри	06.06.2024	*виконано*

Дата видачі завдання 08 квітня 2024р.

Студент (ка_____ Двугрошев А. О.
(підпис)

Керівник роботи _____ ст. викл. кафедри ПІ Новіков Ю. С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, 182 стор., 63 рис., 2 табл., 12 джерел., 10 додатків.

БАГАТОКОРИСТУВАЦЬКІ ІГРИ ДЛЯ ВЕЧІРОК, ІГРОВИЙ ІНТЕРФЕЙС, ІГРОВИЙ ПРОГРАМНИЙ ЗАСТОСУНОК, КЛІЄНТСЬКА ЧАСТИНА, ОСНОВНИЙ ГЕЙМПЛЕЙ, JAVASCRIPT, NODE.JS, REACT, REDUX, SOCKET.IO, TYPESCRIPT, UI/UX

Об’єкт розробки – level-design, основниий хост клієнт та інтерфейс програмного застосунок у жанрі multiplayer party games.

Мета розробки – реалізувати основного екрану клієнтскої частини гри, розробка інтерфейса level-design для трьох ігор у жанрі multiplayer party games які об’єднані програмним застосунком.

Метод рішення – середовище розробки Visual Studio Code, програмна платформа Node.js, бібліотеки React, Socket.IO та Redux, мови програмування JavaScript та TypeScript.

У результаті розробки реалізований core геймплей та інтерфейс основного екрану гри на клієнтській частині системи та розроблений level-design.

MULTIPLAYER PARTY GAMES, GAME INTERFACE, GAMING APPLICATION, CLIENT PART, CORE GAMEPLAY, JAVASCRIPT, NODE.JS, REACT, REDUX, SOCKET.IO, TYPESCRIPT, UI/UX.

The object of development is level-design, the main host client and the interface of software applications in the genre of multiplayer party games.

The goal of the development is to implement the main screen of the client part of the game, the development of a level-design interface for three games in the genre of multiplayer party games, which are united by a software application.

The solution method is the Visual Studio Code development environment, the Node.js software platform, the React, Socket.IO and Redux libraries, the JavaScript and TypeScript programming languages.

As a result of the development, the core gameplay and the interface of the main screen of the game on the client side of the system were implemented, and the level design was developed.

Я, Двугрошев Андрій Олексійович, студент гр. ПЗПІ-20-10, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Ігровий програмний застосунок в жанрі Multiplayer Party Games. Хост частина, левел-дизайн та UI/UX», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

UI – User Interface

UX – User Experience

API – Application Programming Interface

HTTP – HyperText Transfer Protocol

UUID – Universally Unique Identifier

UML – Unified Modeling Language

MVP – Most Valuable Player

QR – Quick Response

ПЗ – програмне забезпечення

ВСТУП

Запит на колективні розваги та спільне проведення часу стає все більшим серед любителів комп'ютерних розваг. Популярність багатокористувацьких ігор постійно зростає, що підтверджується статистикою платформи Steam, де саме такі ігри займають провідні позиції. Це свідчить про постійний попит на можливість спільної гри та взаємодії з іншими гравцями. Ця тенденція вказує на потребу у засобах для спілкування та розваг в онлайн-форматі. Тому розробка ігрових додатків у жанрі multiplayer party games стає надзвичайно актуальною. Сучасні гравці цінують активну взаємодію та веселі розваги з друзями та знайомими, і саме це можуть забезпечити ігри цього жанру. Вони створюють неповторну атмосферу спільної гри та розваг, роблячи зустрічі ще цікавішим.

Темою передатестаційної дипломної роботи є розробка основного хост екрану гри, створення ігрового оточення, розробка UI/UX та level-design у ігровому програмному застосунку у жанрі multiplayer party games. Основне завдання полягає у створенні ефективної системи, що дасть можливість користувачу комфортно взаємодіяти з ігровим програмним застосунком, та розробці ігрового оточення що дасть можливість швидко орієнтуватись у грі та зручно управляти ігровими сценами.

Метою роботи є розробка основної хост застосунку на клієнтській частині системи. Цей компонент відповідатиме за створення ігрового лобі, відображення основного ігрового оточення та геймплею гри, налаштуванню, навігації та вибору гри у програмному застосунку. Крім того, метою є створення UI/UX та побудову level-design, яке дасть можливість зручно орієнтуватись у ігровому процесі та управляти ним. Для досягнення цих цілей використовуються такі технології, як середовище розробки Visual Studio Code, програмна платформа Node.js, бібліотеки React та Socket.IO, а також мови програмування JavaScript.

Ігровий програмний застосунок у жанрі multiplayer party games може бути використаний для розваг групи людей від 4 до 8 гравців. Для організації ігрового

процесу необхідний лише доступ до Інтернету, веб-браузер, персональний комп’ютер з монітором для створення ігрової сесії та відображення оточення, а також смартфони для використання мобільного ігрового контролера для гравців, які будуть підключатися до ігрової сесії. Застосунок має бути привабливим через свою простоту та доступність, адже для початку гри немає необхідності встановлювати нічого, окрім веб-браузера. Це робить його гарним вибором для широкого кола аудиторії, а також для різних заходів, від онлайн гри з друзями до особистих зустрічей гравців.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Ігри жанру multiplayer party games призначені для групового відпочинку, часто на вечірках або зустрічах друзів. Вони спрямовані на спільну гру, змагання або просто веселощі, мають прості правила і механіку, що дозволяють всім приєднатися. Цей жанр стає все популярнішим завдяки соціальній активності в інтернеті і здатності об’єднувати людей для спільного часу. Інновації у розвитку цих ігор, з урахуванням поточних тенденцій і потреб користувачів, обіцяють успіх на ринку. Розвиток технологій, штучного інтелекту та віртуальної реальності також сприятиме росту цього сегменту ринку.

Розроблений ігровий додаток складається з клієнтської та серверної частин. Клієнтська частина ділиться на сторінки для відображення геймплею та сторінки для контролю гри за допомогою смартфону. Серверна частина за передачу даних до клієнта, стабільну роботу системи, обробку та управління ігровою сесією.

Цільова аудиторія гри жанру multiplayer party games зазвичай охоплює людей різного віку. Здебільшого це чоловіки віком від 14 до 35 років, які цінують конкуренцію, логіку та стратегічні аспекти гри. Запуск застосунку через браузер та легкість механік дозволяю максимально розширити спектр гравців, які шукають якісний та захоплюючий геймплей.

Мета ігрового застосунку полягає у створенні захоплюючого досвіду для користувачів при грі в нього гравців, які хочуть спільно провести час та розважитися. Основними цілями цієї системи є:

– забезпечити інтуїтивність та зрозумілість інтерфейсу для широкого кола гравців;

– розробка цікавих та різноманітних ігрових механік, які будуть сприяти активній взаємодії між гравцями;

– забезпечити безпечність та надійність мережевого підключення між гравцями, для безперебійної гри в режимі реального часу з можливістю повторне підключитися до ігрової сесії;

– забезпечити можливість використовувати застосунок повністю через інтернет браузер, для повної доступності та зручності для широкого кола гравців.

З цього випливає, що мета програмного ігрового застосунку – розробити програмне забезпечення, яке привертає увагу багатьох гравців, заохочує їх активну участь, має добре налаштовану мережеву взаємодію та є доступним для всіх користувачів системи.

Фрагменти предметної області з розробки проєктного ігрового програмного застосунку включають:

а) для сервера:

1) швидке отримання та обробка даних від клієнту;

2) зберігання та управління даними гри;

3) стабільна робота мережі та підтримка з’єднання між основним екраном і контролерами.

4) генерація ігрового ключа для підключення до існуючих ігрових сесій;

б) для клієнтської частини контролера:

1) отримання та обробка даних з сервера;

2) видача даних до сервера з мінімальною затримкою;

3) відображення поточного стану гри та можливість взаємодії з нею через веб-браузер на смартфоні або іншому пристрої;

4) введення ігрового ключ для підключення до існуючих ігрових сесій;

в) для клієнтської частини екрану ігрового оточення:

1) вибір гри та видача даних для генерації лобі сервером;

2) відображення інформації про поточний етап гри;

3) обробка отриманих даних з сервера;

Перераховані фрагменти описують різноманітні завдання, пов’язані з управлінням даними: зберіганням, обробкою, виведенням і забезпеченням взаємодії користувачів системи між собою та з сервером.

Інформаційні потреби предметної області з розробки проєктного ігрового програмного застосунку включають:

– отримання інформації про поточний стан гри на основному хост екрані ігрового оточення та на контролерах;

– отримання інформації команд користувача, які потрібно виконати в певний момент гри на основному хост екрані ігрового оточення та на контролерах;

– отримання інформації про конкретну гру перед її вибором на основному екрані гри;

– отримання інформації про ігрове лобі при підключенні до неї;

– отримання інформації про кінцевий результат гри на екрані ігрового оточення та на контролерах.

Користувачі мають доступ до важливої інформації про міні-ігри, процес гри, взаємодію з іншими гравцями, результати та досягнення, що допомагає створити задоволений і зрозумілий ігровий досвід.

Перший аналог – це веб-застосунок «Gartic Phone», розроблена студією «Onrizon Social Games». Основна відмінність «Gartic Phone» від інших ігор у жанрі multiplayer party games, мультиплатформеність та повний запуск через браузер, без додаткових завантажень на комп'ютер або смартфон, унікальні механіки де кожен учасник має змогу вносити свій внесок до кожної з варіантів ігор.

Після відкриття веб-додатку https://garticphone.com/, користувачі потрапляють на екран головного меню (див. рис. 1.1). Тут можна переглянути інформацію про соціальні мережі розробників, ввести свій нікнейм, переглянути правила гри, обрати аватар та створити ігрове лобі за допомогою кнопки «Start».

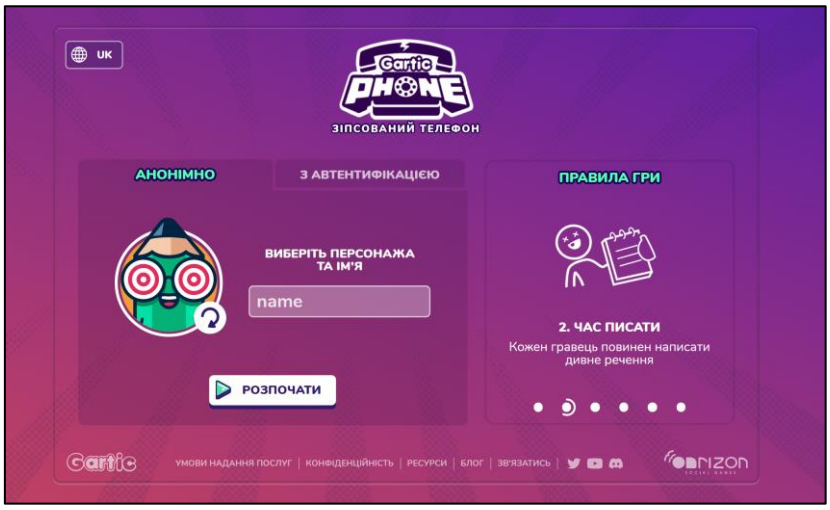


Рисунок 1.1 – Меню підключення до приватної ігрової кімнати на веб-сайті https://garticphone.com/ (знімок екрана виконано самостійно)

Після створення приватного лобі, користувач потрапляє на екран (див. рис. 1.2), де він може додати гравців за допомогою генерування посилання кнопкою «Invite», також можна обрати режим гри, та кількість гравців.

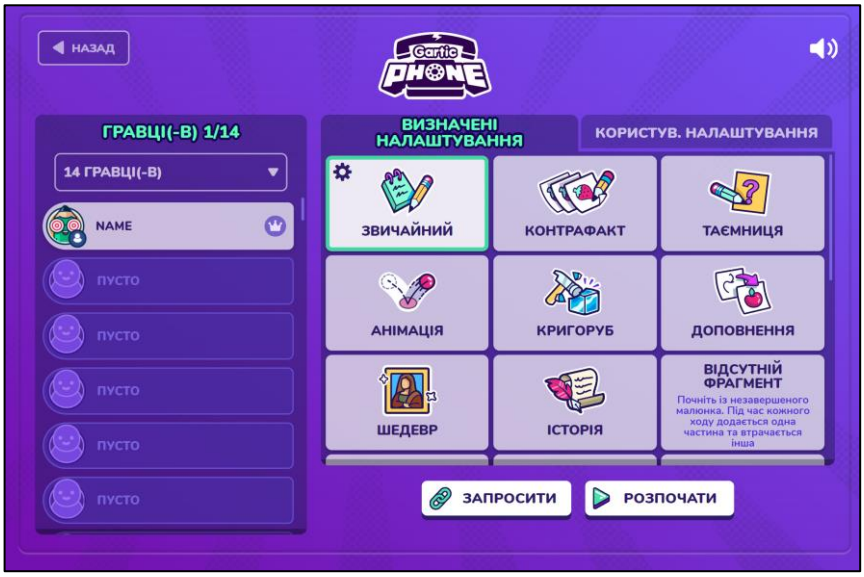


Рисунок 1.2 – Приватне лобі на веб сайті https://garticphone.com/ (знімок екрана виконано самостійно)

В залежності від обраного режиму, користувачу потрібно розробити малюнок або написати історію до нього, загалом ці дві механіки є основою всього програмного застосунку. На початку гри користувач бачить екран (див. рис. 1.3), опису малюнку.

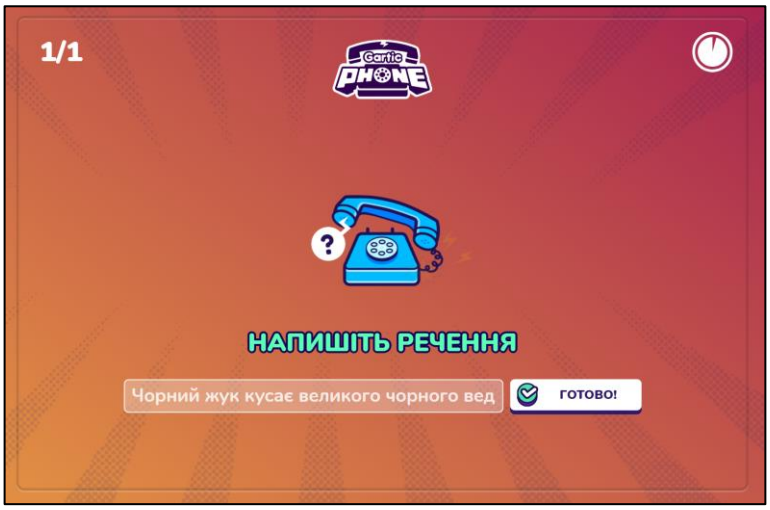


Рисунок 1.3 – Опис малюнку на веб сайті https://garticphone.com/ (знімок екрана виконано самостійно)

Після опису малюнків гравці приступають до намалювання рисунку (див. рис 1.4) в залежності від завдання або теми яка випала гравцям. Користувачі бачать тему малюнку, полотно та інструменти для малювання.

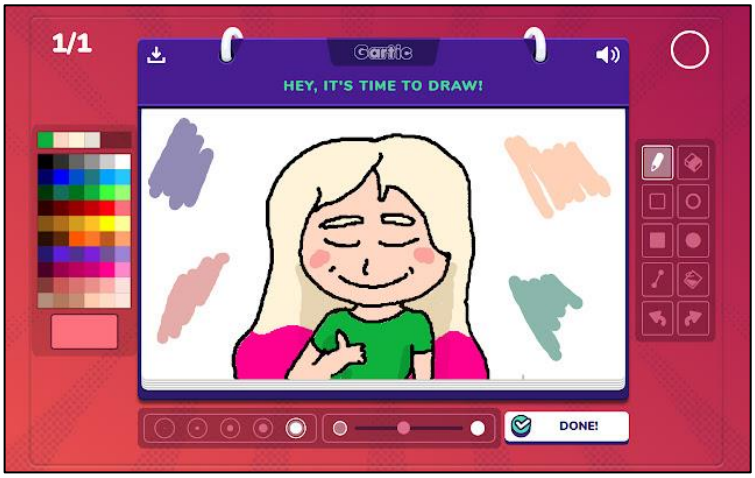


Рисунок 1.4 – Екран з завданням на веб сайті https://garticphone.com/ (знімок екрана виконано самостійно)

До переваг цієї гри можна віднести наступне:

– можливість додавання до ігрової сесії великої кількості гравців;
– інтуїтивний та простий інтерфейс;
– можливість збереження створених малюнків в PNG або GIF форматі;

– велика кількість мов перекладу.

– безкоштовність;

– повністю працююча браузерна версія гри;

– зручне створення аккаунту.

До недоліків цієї гри можна віднести наступне:

– інтегровану озвучку неможливо регулювати по гучності.

– однотипність режимів та ігрових механік, всі вони відносяться або до малювання або до написання тексту;

– мала кількість кольорів для малювання.

Другий аналог – серія ігор «The Jackbox Party Pack», розроблена студією «Jackbox Games». Одна з основних відмінностей «The Jackbox Party Pack» від інших ігор у цьому жанрі полягає в унікальних різноманітних механіках, які міняються залежно від гри, та системи мобільних контролерів на смартфонах, за допомогою яких гравці можуть грати в гру.

Після завантаження гри на свій пристрій гравець може, змінити налаштування гри, подивитись опис ігор та обрати одну з запропонованих через головне меню (див. рис.1.5).

Рисунок 1.5 – Головне меню «The Jackbox Party Pack» (знімок екрана виконано самостійно

Після вибору гри, створюється приватне лобі з унікальним кодом (див. рис. 1.6). На цьому екрані відображається підключені гравці та інформація щодо початку гри.

Рисунок 1.6 – Приватна кімната «The Jackbox Party Pack» (знімок екрана виконано самостійно)

Через свої пристрої гравці відкривають веб-браузер та переходять на веб-сайт https://jackbox.tv/, де їх зустрічає меню для підключення до приватної кімнати (див. рис 1.7). Тут можна переглянути загальну інформацію про компанію та їх продукти, побачити список доступних пакетів, ввести код кімнати та свій нікнейм, а потім натиснути кнопку «Play», щоб підключитися до ігрової сесії.

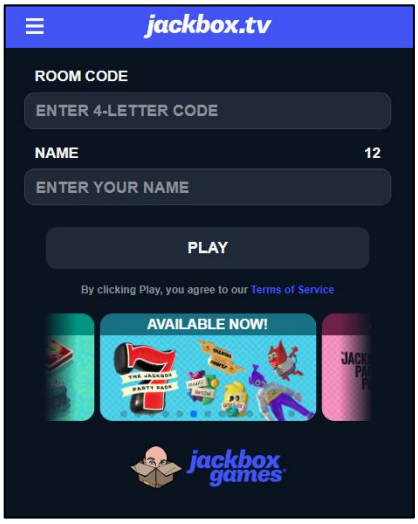


Рисунок 1.7 – Меню підключення до приватної кімнати на веб-сайті https://jackbox.tv/ (знімок екрана виконано самостійно)

На прикладі міні-гри «Joke ship», на екрані основного ігрового оточеня (див. рис. 1.8) відображаються гравці, що беруть участь у раунді, а також ігрова ситуація до якої потрібно придумати жарт.

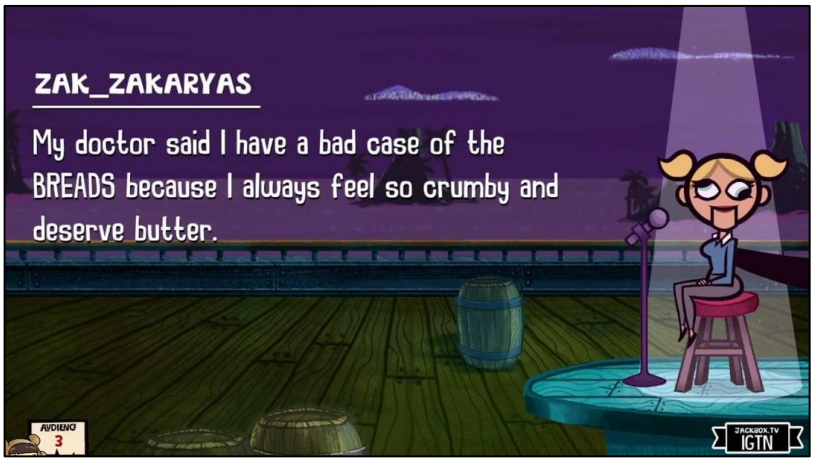


Рисунок 1.8 – Екран ігрового оточеня «The Jackbox Party Pack» з запущеною міні-грою (знімок екрана виконано самостійно)

Після оголошення ситуації на екрані контролерів гравців (див. рис. 1.9) з’являється елементи управління, для додавання жарту або іншої унікальної механіки залежності від гри.

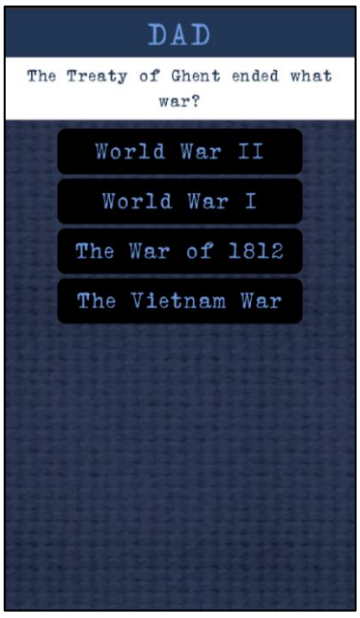


Рисунок 1.9 – Екран контролера користувача на веб-сайті https://jackbox.tv/ (знімок екрана виконано самостійно)

До переваг цієї гри можна віднести наступне:

– професійна акторська озвучка;
– унікальний дизайн;
– унікальний варіант управління грою;
– простота використання;
– різноманітність ігрових механік;
– приємні анімації та інтерфейс користувача.

До недоліків цієї гри можна віднести наступне:

– обмеженість до перекладів деяких ігор;
– гра вимагає завантаження та встановлення на пристрій користувача.

Після аналізу конкурентів можна зробити висновок, що успішність залежить від того, наскільки продукт відрізняється від інших, враховуючи їх переваги і недоліки. Необхідно розробити унікальні механіки, які інші продукти ще не пропонували та унікальні рішення для збільшення аудиторія, наприклад розташування гри в браузері.

1.2 Виявлення та вирішення проблем

З інноваціями в галузі технологій та розширенням доступу до Інтернету, ігрова індустрія постійно шукає нові способи привернення гравців і створення захоплюючих ігрових вражень В цьому контексті жанр multiplayer party games обирає все більшої популярності. Ці ігри зазвичай відрізняються від традиційних ігор за своїм форматом, який орієнтований на груповий досвід. Вони можуть включати різноманітні міні-ігри, елементи співпраці або конкуренції, а також можливості для інтерактивного спілкування між гравцями. Це робить їх ідеальними для проведення вечірок чи віртуальних зустрічей з друзями.

Зі зростанням аудиторії ігор у такому жанрі, зростає конкуренція та потреба у вдосконаленні ігрових застосунків. Ключовим завданням стає виявлення та вирішення таких проблем, що в свою чергу удосконалює досвід для гравців. Проєктування простих але в той же час унікальних та цікавих механік забезпечить покращення ігрового досвіду як для досвідчених так і для нових гравців. Велике різноманіття ігрових механік у жанрі multiplayer party games разом з використанням контролера в веб-застосунку на мобільному пристрої привернуть увагу великої кількості нових користувачів. Використання смартфону надасть можливість почати гру з умовою підключення до Інтернету та додаткового екрану де буде виводитись основне ігрове оточення для всіх гравців. Робота додатка у браузері дозволить користувачам мінімізувати підготовчий етап перед ігровою сесією, та дасть можливість почати грати в застосунок одразу при необхідності.

Вирішення цих проблем покращить ігровий досвід, та забезпечить розширення можливого обсягу зацікавлених гравців. Інтеграція мобільного телефона до гри надасть можливість додавати до застосунку унікальні механіки, які підвищать конкурентоспроможність на ринку ігор в жанрі multiplayer party games.

1.3 Постановка задачі

Щоб задовольнити потреби користувачів, які бажають соціальної взаємодії та активно проводити час в спільній грі з друзями, потрібно розробити ігровий застосунок у жанрі multiplayer party games. На початковому етапі розробки цей застосунок буде включати три міні-ігри, які можна вибрати з головного меню хоста: «Brain Knights», «Turing Test» та «Skibidy Party».

«Turing Test» – це командна соціальна дедуктивна гра, розрахована на участь від чотирьох до восьми гравців та складається з трьох раундів. Гравці діляться на дві команди: студенти і професори за відношенням: чотири гравці – один студент

та три професори; п’ять гравців – два студенти та три професори; шість гравців – два студенти та чотири професори; сім гравців – три студенти та чотири професори; вісім гравців – три студенти та п’ять професорів.

Гра належить до категорії ігор з соціальним маскуванням, до таких відноситься гра «Мафія», головною ідеєю гри «Turing Test» маскування гравців які грають за професорів під додаткового гравця який є штучним інтелектом, та спробувати ввести в оману команду гравців студентів, які в свою чергу будуть наводити питання, які можуть видати штучний інтелект поміж гравців.

Команда студентів починає гру з того, що кожний гравець задає своє питання до протилежної команди, після цього протилежна команда відповідає на ці питання, штучний інтелект отримує питання та всі відповіді та генерує свій замаскований під гравців варіант, після цього команда студенів може передивитись по черзі питання яка задала команда, та відповіді до них. Методом голосування до кожного питання, команда студентів визначає хто є бот, і в залежності від правильності обраного варіанту командам буде надано одне або два бали. Перемагає та команда, яка набрала більше очок. Якщо ж бали розподілилися порівну, гра закінчується в нічию.

«Skibidy Party» – це гумористична карткова гра, розрахована на участь від чотирьох до восьми гравців, що складається з кількості раундів, рівної кількості учасників гри.

Гра націлена на протистояння гравців один одному. Кожен раунд випадковим чином обирається суддя, який зачитує унікальну жартливу ситуацію. У кожному раунді інші гравці отримують по шість карток з мемами. Після того, як суддя зачитує ситуацію яку гра вибрала з ситуаційної колоди карток, гравці вибирають найкумедніший по їх думці мем з шести карток які були видані на початку раунду. Після цього суддя отримує список мемів, обраних гравцями, і розставляє їх по місцях від першого до третього. За перше місце учасник отримує 1500 очок, за друге – 1000 очок, за третє – 500 очок. Перемагає той, хто набрав найбільшу кількість балів протягом усіх проведених раундів.

«Brain Knights» – це командна змагальна вікторина, розрахована на участь від чотирьох до восьми гравців і складається з двох етапів. Команди складаються з рівної кількості гравців, якщо гравців парна кількість, або на один менше в будь-якій команді. У кожної команди є свій капітан, який міняє назву команди, обирає гравця який буде змагатися з іншим я обраній темі. На першому етапі гри відбувається послідовність раундів. Кількість раундів визначається як подвійне число максимальної кількості гравців у будь-якій з двох команд, з метою того, щоб кожен учасник зіграв принаймні двічі у цьому етапі.

На початку кожного раунду командири обирають гравців для змагання, після цього починається бліц-опитування, хто набере більшу кількість балів, той і отримає бал до командного заліку. Якщо кількість правильних відповідей однакова, то обидва учасники приносять своїм командам по одному зароблену очку. Під кінець першого етапу визначається найефективніший гравець –той, хто надав найбільшу кількість правильних відповідей. Другий етап – це командна вікторина, в якій беруть участь всі члени команди. Етап складається з трьох раундів та складається з більш складних командних питань. На початку такого раунду капітани методом вилучення обирають тему на яку вони отримають три питання, після цього йде етап відповідей, де всі гравці на відмінну від першого раунду, приймають участь у виборі правильної відповіді. У випадку однакової кількості балів визначається нічия.

Програмний застосунок поділяється на дві частини:

– серверна частина: відповідає за управління, зберігання та обробку даних, та витупає шлюзом з’єднання клієнтської частини де відображається ігровий процес та клієнтської частини контролера для смартфону;

– клієнтська частина: в свою чергу ділиться на екран з основним відображенням геймплея та мобільний контролер, перша частина відповідає за вивід геймплею на екран, вибір гри, створення та керування ігровими лобі, отримання даних з сервера. Частина контролера відповідає за надання команд гравця до системи, додавання тексту, зміну персонажу, відправку даних до сервера.

Необхідно розробити серверну частину за допомогою TypeScript та Node.js, а також клієнтський веб-застосунок з використанням JavaScript та React. Для забезпечення реального часу обміну даними між клієнтською та серверною частинами планується використання бібліотеки WebSocket на сервері та клієнтській частині. Головною метою є забезпечення доступності ігрового застосунку з будь-якого пристрою з Інтернет-підключенням та веб-браузером.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Основною метою цієї програмної системи є створення якісного програмного застосунку з зрозумілим інтерфейсом у програмному застосунку жанру multiplayer party games. Система пропонує користувачам новий та неповторний ігровий досвід за допомогою розроблених міні-ігор та впроваджених інновацій, а також забезпечує зручний, інформативний та зрозумілий геймплей. Такий підхід забезпечує зацікавленість та задоволення гравців від ігрового процесу, а також дозволяє конкурувати на ринку ігрових програмних застосунків у цьому жанрі.

Ігровий програмний застосунок у жанрі multiplayer party games повинен вирішувати наступні завдання:

– створення інтуїтивного та якісного інтерфейсу та ігрового оточення.
– створення приватної кімнати для кожної ігрової сесії;
– розробка інструментів для зручного керування ігровим процесом;
– обробка та відображення поточної ігрової активності.

Ігровий програмний застосунок у жанрі multiplayer party games має наступні функціональні вимоги:

а) хост частина клієнту:

1) загальне:

– забезпечення зручної навігації зі збереженням інформації про останню обрану міні-гру;

– вибір однієї з наявних міні-ігор на головному екрані («Brain Knights», «Turing Test» або «Skibidy Party»);

– створення приватної ігрової кімнати для кожної ігрової сесії;

– управління поточним станом розпочатої ігрової сесії;

– відображення інформації про створену приватну кімнату для можливості підключення до неї;

– відображення часу, що залишився до зміни етапу гри;

– відображення гравця, який є власником ігрової сесії;

– відображення нікнеймів, стану підключення та готовності кожного з гравців;

– зміна поточного екрану відповідно до даних, що були отримані з сервера;

– валідація даних, що відправляються на сервер;

2) міні-гра «Brain Knights»:

– відображення приналежності кожного гравця до певної команди та їх капітанів;

– відображення аватарів гравців та назв команд;

– відображення кількості правильних відповідей, які надав кожен гравець;

– відображення теми вікторини та її запитань;

– відображення правильних та неправильних відповідей;

– налаштування логіки відображення правильних відповідей гравців (на першому етапі вони виводяться одразу після відповіді одного з учасників, а на другому – після відповідей усіх капітанів команд);

– відображення аватарів гравців або команди поруч із обраним варіантом відповіді;

– відображення переможця раунду;

– відображення загального рахунку команди;

– відображення MVP-гравця першого етапу гри;

– відображення тем для виключення та інформації про черговість;

– відображення результату гри;

– відображення титулів гравців у кінці гри;

3) міні-гра «Turing Test»:

– відображення приналежності кожного гравця до певної команди;

– відображення інструкцій щодо виконання завдань;

– відображення поставлених запитань та наданих відповідей;

– відображення наданих голосів за варіанти відповідей, які, на думку студентів, надав бот;

– відображення відповіді, яку надав бот, а також результату голосування команди студентів;

– відображення переможців та проміжних результатів раундів;

– відображення результату гри;

– відображення титулів гравців у кінці гри;

4) міні-гра «Skibidy Party»:

– завантаження власного набору мемів до ігрової сесії;

– валідація завантажуваних файлів;

– відображення аватарів та ролей гравців;

– відображення обраної ситуації;

– відображення мемів, які обрали гравці;

– відображення переможців кожного раунду;

– відображення очок, які заробив кожен гравець за раунд;

– відображення мемів, які обрали гравці;

– відображення результату гри;

– відображення титулів гравців у кінці гри;

б) створення оточення та UI/UX:

1) створення повного макету застосунку за допомогою Figma;

2) налаштування оточення так щоб воно було інтуїтивне та просте в використанні користувачем.

Ігровий програмний застосунок має наступні нефункціональні вимоги:

а) продуктивність:

– програма повинна працювати з мінімальним часом відгуку від системи;

– час завантаження ігрових сесій та переходів між екранами має бути мінімальним;

б) масштабованість:

– архітектура програми повинна бути гнучкою та розширюваною, щоб забезпечити можливість додавання нових ігрових режимів або функцій у майбутньому без значних змін у коді;

– застосунок має мати можливість легко масштабуватися для підтримки великої кількості одночасних ігрових сесій і гравців;

в) надійність:

– система має гарантувати, що користувачі зможуть отримати доступ без перебоїв, навіть у випадку непередбачених обставин або помилок;

г) сумісність:

– система повинна підтримувати стабільну роботу в усіх веб-браузерах;

д) локалізація:

– зміна локалізації має відбуватись для всіх учасників гри на контролерах та екрані оточення;

– інтерфейс програмного застосунку повинен бути реалізований українською та англійською мовами.

Основний екран ігрового оточення буде реалізований на клієнтскій частині гри у вигляді веб-застосунку для пристроїв з великими екранами, використовуючи мову програмування JavaScript та бібліотеку React [1-2]. Інтеграція штучного інтелекту буде виконана на серверній частині, використовуючи мову програмування TypeScript та платформу Node.js. Технології, що будуть використовуватися: Node.js, React, Nest.js, I18next, JavaScript, TypeScript, Socket.IO, Redux, HTML [3-4].

Розробка ігрового програмного застосунку буде здійснюватися у кілька етапів:

а) підготовчий етап:

1) створення і визначення системних вимог;

2) планування програмної архітектури;

3) конфігурація робочого середовища та підготовка інструментів;

б) розробка клієнтської частини:

1) моделювання ігрового оточення;
2) налаштування запитів;
3) налаштування локального сховища даних;
4) налаштування маршрутизації клієнтської частини;
5) налаштування перекладу;
6) реалізація геймплею;
7) верстка модулів оточення;
8) об’єднання модулів оточення та геймплею;

в) тестування та оптимізація:

1) виконання ручного тестування;
2) покращення продуктивності та виправлення виявлених недоліків.

Для розроблюваного ігрового застосунку було створено специфікацію (див. додаток В), яка містить загальний опис проєкту та його конкретні вимоги. Розробка гри має чітко сплановані етапи, які враховують взаємозв'язок процесів. Це дозволяє ефективно використовувати ресурси і гарантує завершення проєкту вчасно та у встановлені терміни.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

Для розроблюваного ігрового програмного застосунку в жанрі multiplayer party games було створено діаграму розгортання (див. рис. 3.1).

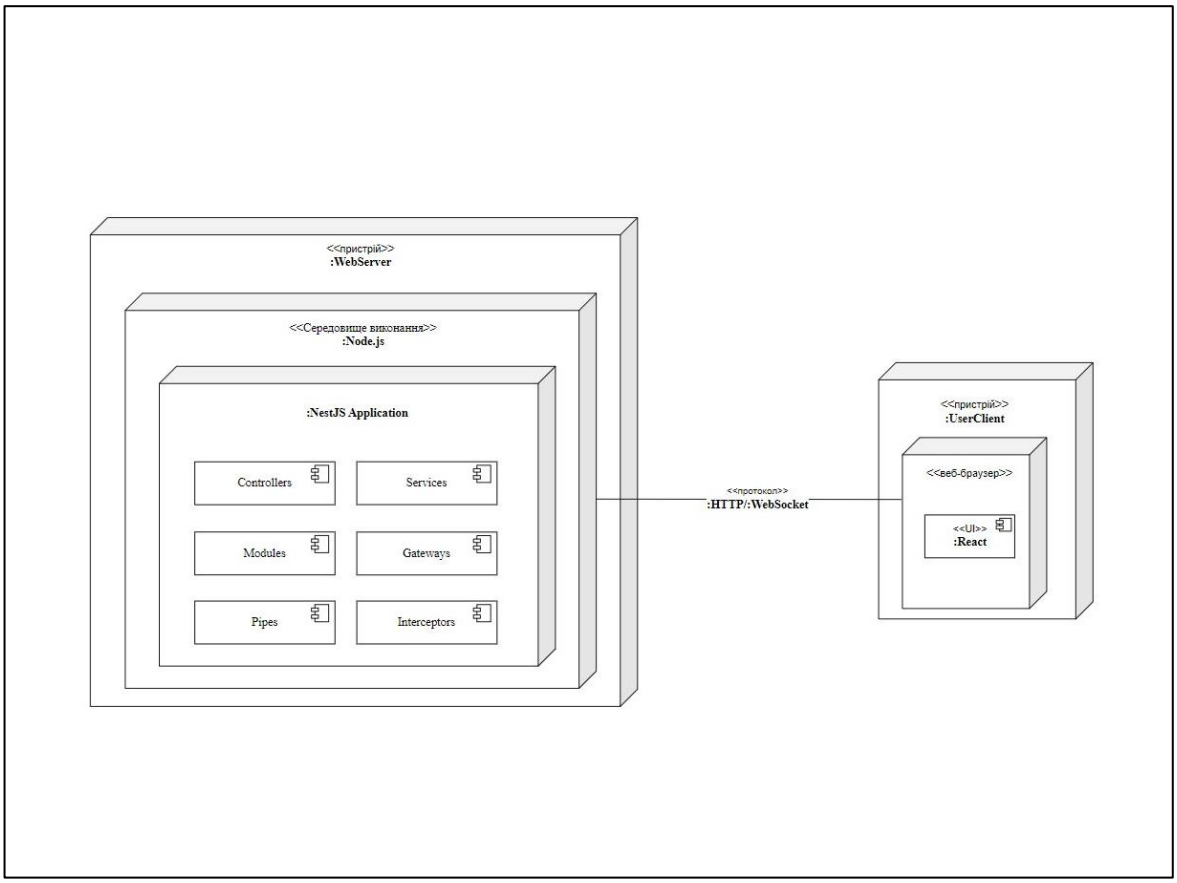


Рисунок 3.1 – UML діаграма розгортання ігрового програмного застосунку (діаграма виконана самостійно)

Система складається з двох основних частин: веб-сервера, який забезпечує надійну роботу мережі, маршрутизацію запитів та збереження даних сеансів гри, і веб-застосунку, який надає зручний інтерфейс для взаємодії з системою через веб-браузер, керує грою, відображає інформацію та передає дані на сервер.

Для розроблюваного core геймплею на клієнтській частині ігрового програмного застосунку в жанрі multiplayer party games було створено діаграму прецедентів (див. рис. 3.2), діаграму компонентів (див. рис. 3.3), діаграму пакетів (див. рис. 3.4), діаграму взаємодії (див. рис. 3.5), діаграму діяльності (див. рис. 3.6) та діаграму станів (див. рис. 3.7-3.10).

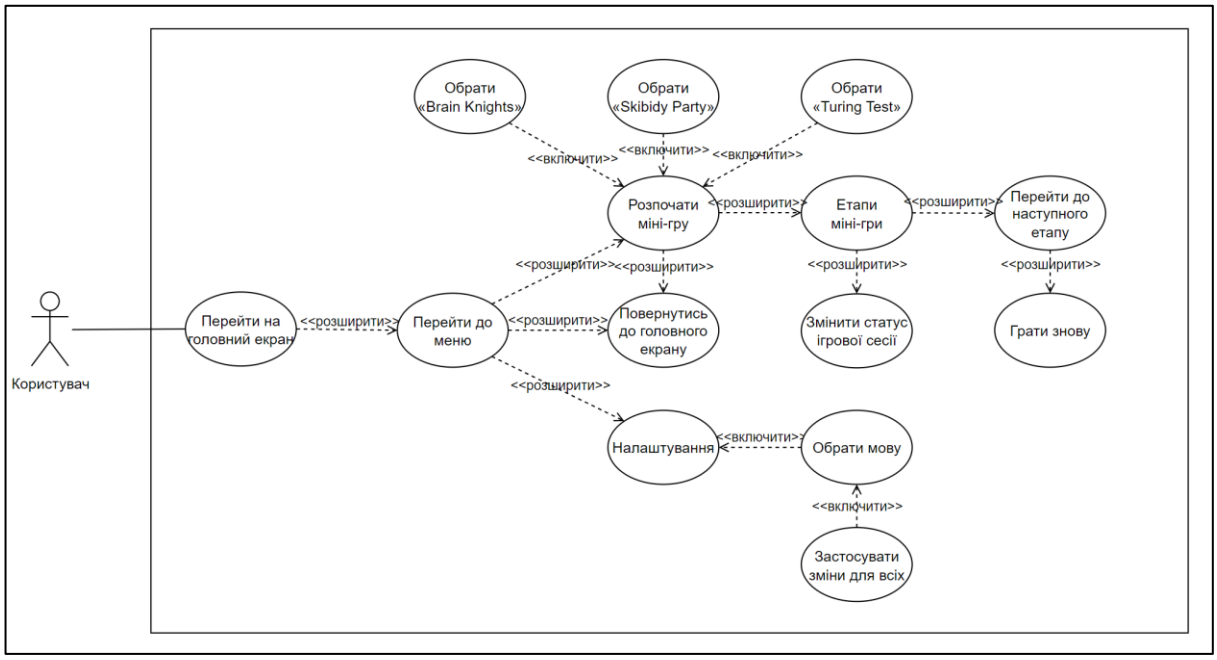


Рисунок 3.2 – UML діаграма основного ігрового середовища на клієнтській частині ігрового програмного застосунку (діаграма виконана самостійно)

Користувач може взаємодіяти з системою наступним чином:

– налаштовувати гучність звуку;
– налаштовувати локалізації;
– розпочинати сесію гри;
– змінити статус ігрового лобі;
– користуватися навігаціє;
– обрати одну з трьох доступних міні-ігор;
– грати у доступні міні-ігри повторно.

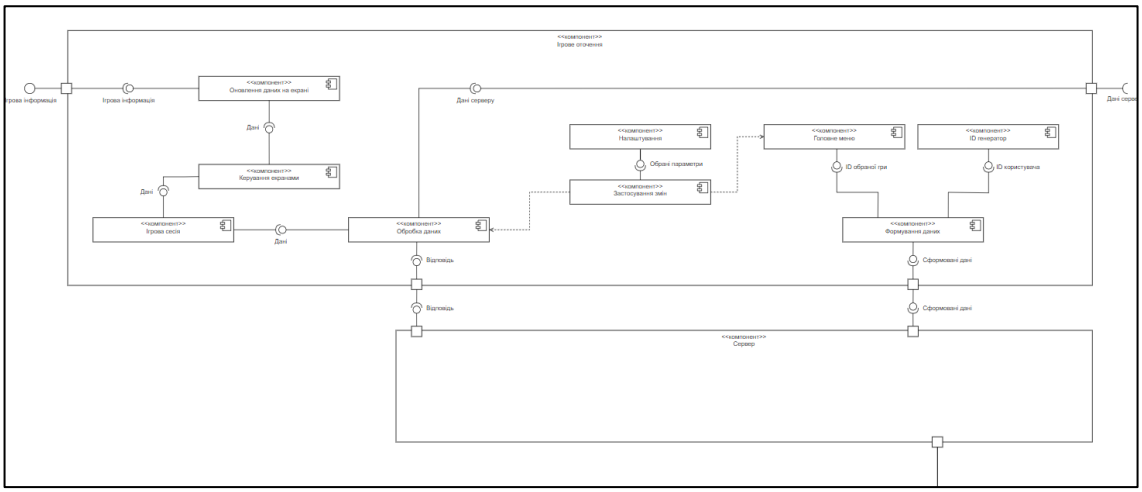


Рисунок 3.3 – перша частина UML діаграма компонентів клієнтській частині ігрового програмного застосунку (діаграма виконана самостійно)

Перша частини діаграми компонентів клієнтської частини відображає взаємозв’язок між клієнтською частиною з основним ігровим середовищем та сервером програмного застосунку. Ці компоненти описують взаємодію клієнтської частини з сервером, відправку та отримання даних з частин програмної системи, навігація та керуванням ігровим процесом.

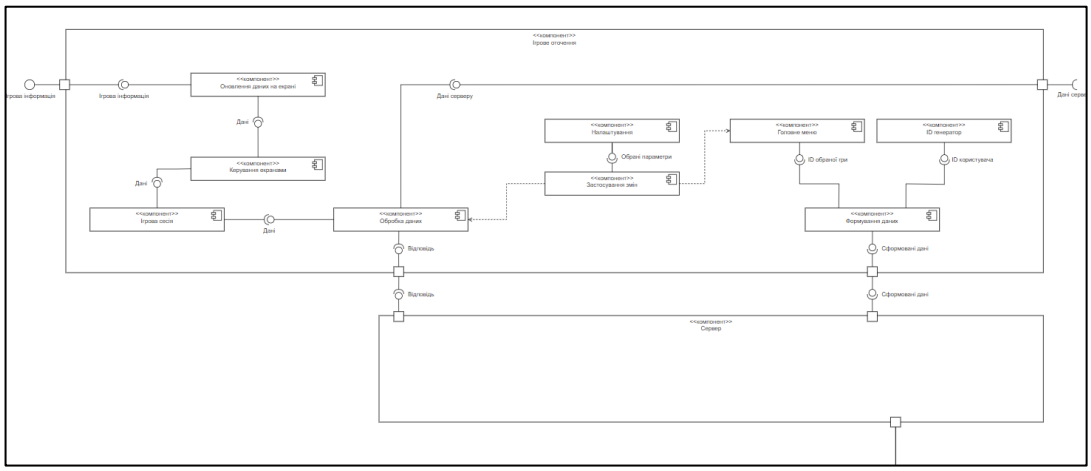


Рисунок 3.4 – друга частина UML діаграма компонентів клієнтській частині ігрового програмного застосунку (діаграма виконана самостійно)

Друга частини діаграми компонентів клієнтської частини відображає взаємозв’язок між мобільним браузерним контролером та середовищем та сервером програмного застосунку. Ці компоненти описують взаємодію мобільного

ігрового контролера з сервером, контроль ігрового процесу користувачем, механіку підключення до ігрової сесії.

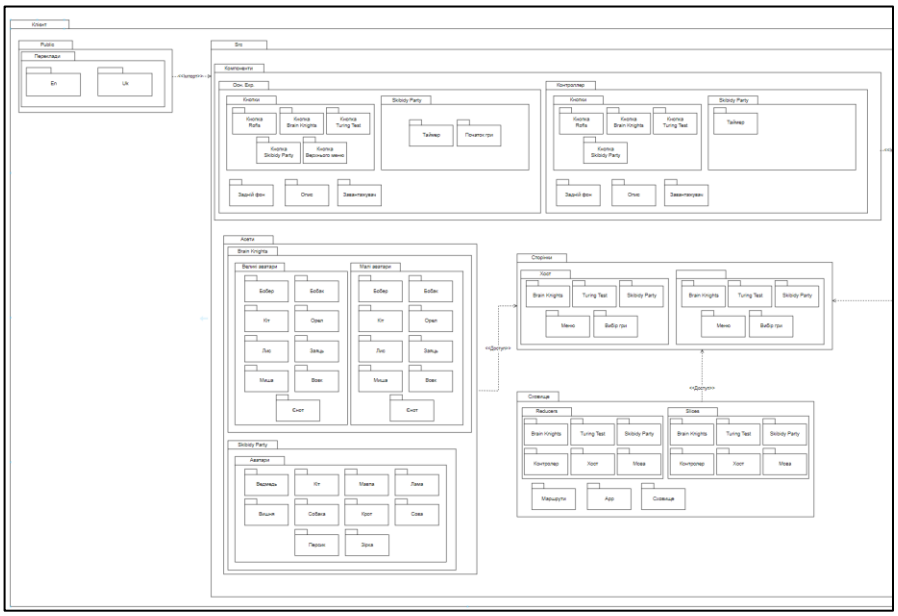


Рисунок 3.5 – Перша частина UML діаграми пакетів клієнтської частини ігрового програмного застосунку (діаграма виконана самостійно)

Перша частина діаграми компонентів, в якій описується розташування сторінок, ассетів, та набору даних сховища.

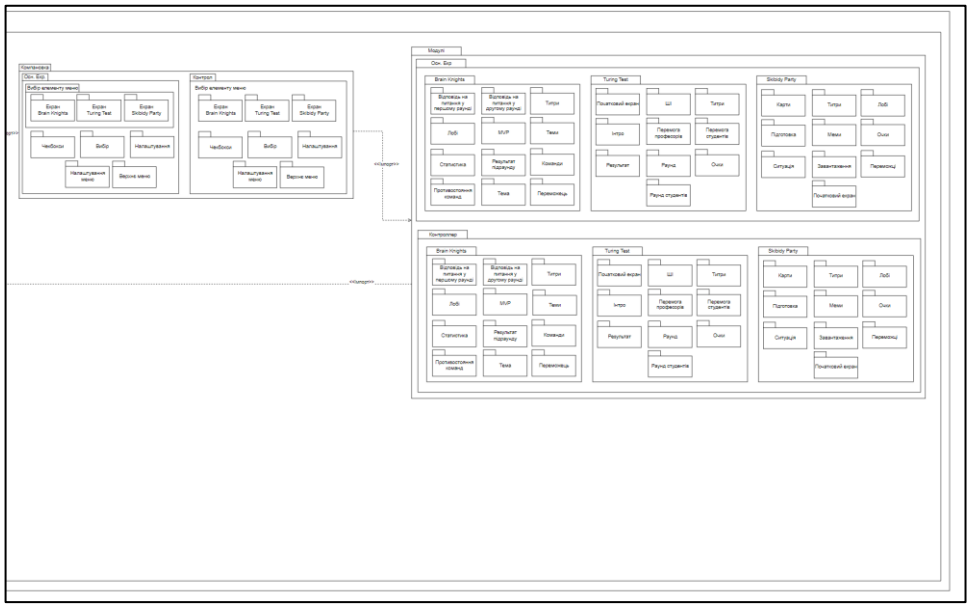


Рисунок 3.6 – Друга частина UML діаграми пакетів клієнтської частини ігрового програмного застосунку (діаграма виконана самостійно)

Друга частина діаграми описує розташування ресурсів основних сторінок хост частини проєкту та контроллерів у всіх створених іграх.

Діаграма пакетів клієнтської частини відображає організацію та розташування елементів та модулів у проєкті [5]. На ній зображено як структуруються дані у екрані ігрового оточення та мобільному ігровому контролері, які програмні ресурси поділені та ізольовані, які використовують модулі разом.

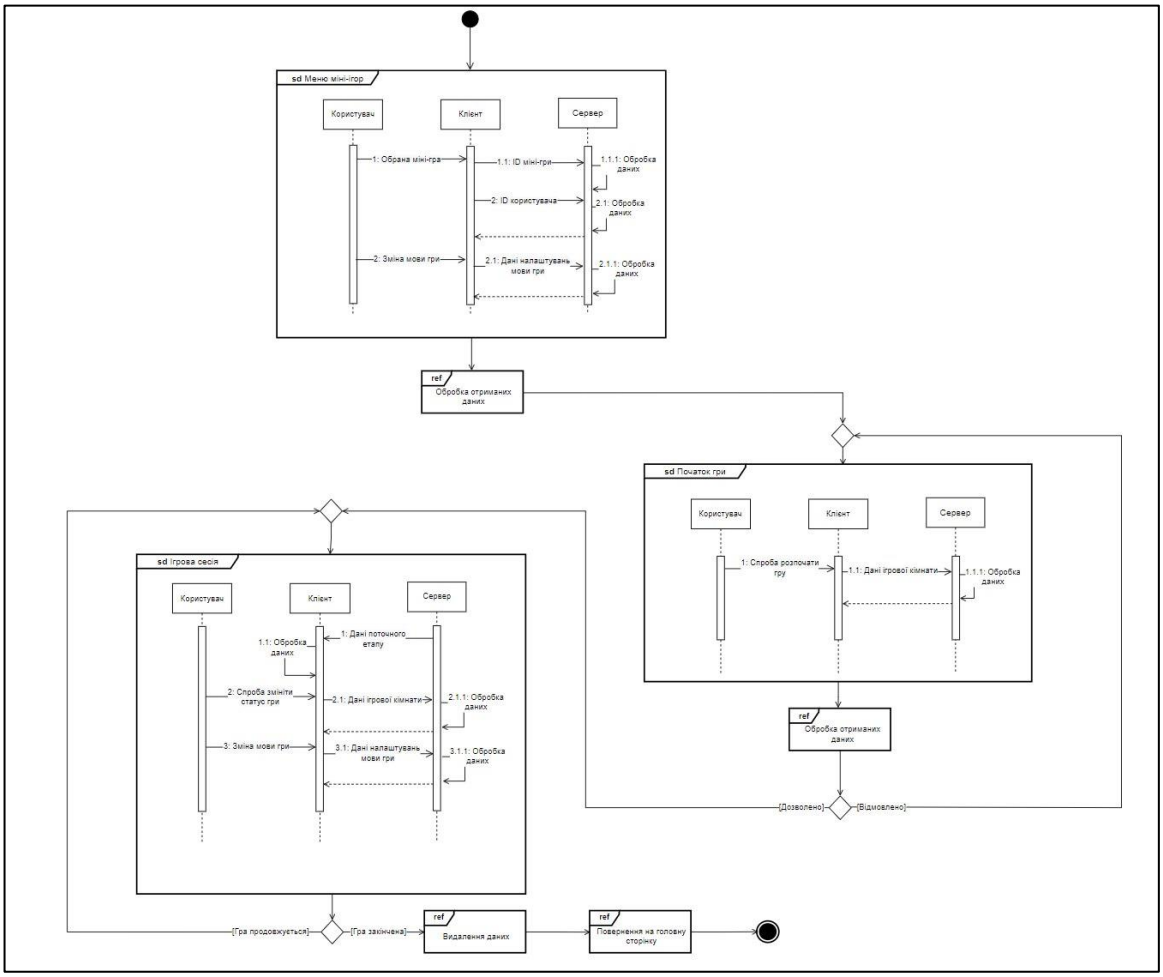


Рисунок 3.7 – UML діаграма взаємодії core геймплею на клієнтській частині ігрового програмного застосунку (діаграма виконана самостійно)

Діаграма взаємодії клієнтської частини зосереджується на описі потоку повідомлень всередині системи. На ній відображено послідовність логіки розробленого застосунку та взаємодію між акторами та компонентами системи.

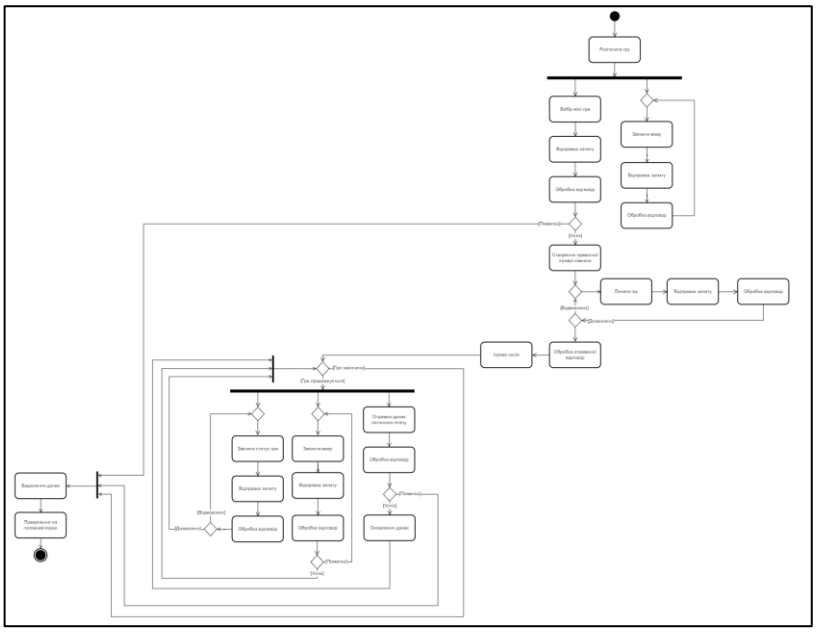


Рисунок 3.8 – UML діаграма діяльності core геймплею на клієнтській частині ігрового програмного застосунку (діаграма виконана самостійно)

UML діаграма діяльності екрану ігрового оточення клієнтської частини відображає послідовну реалізація логіки системи, як дії в застосунку переходять від однієї до іншої на основному екрані.

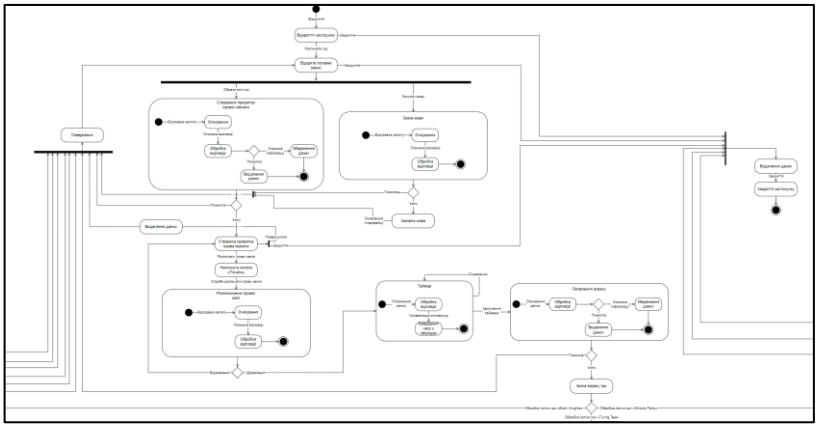


Рисунок 3.9 – UML діаграма станів екрану ігрового оточення, до початку гри, на клієнтській частині ігрового програмного застосунку (діаграма виконана самостійно)

Перша частина UML діаграми описує обробку логіки початку ігрової сесії. Це створення ігрової сесії, налаштування клієнту, налаштування гучності звуку,

налаштування локалізації, механіка ігрового таймеру, оновлення ігрового оточення, навігація та вибір гри з пропонованого списку.

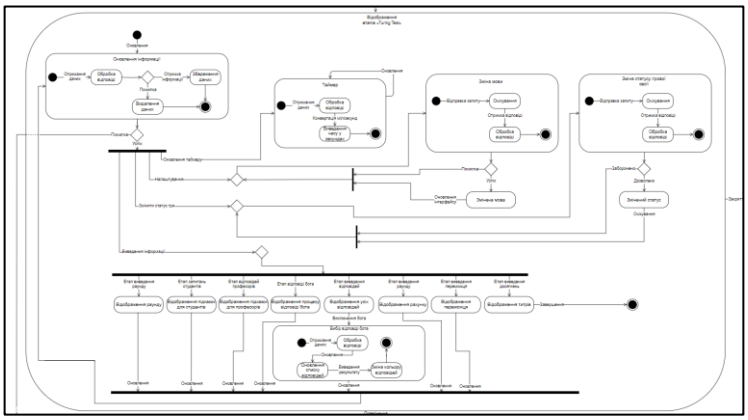


Рисунок 3.10 – UML діаграма станів екрану ігрового оточення, міні-гри «Turing Test», на клієнтській частині ігрового програмного застосунку (діаграма виконана самостійно)

Друга частина UML діаграми описує обробку логіки гри «Turing Test», а саме: відображення таймеру, збереження та видалення ігрових даних, зміна мови та гучності звуку, зміна статусу сесії, механіка паузи, відображення поточного раунду, відповідей, зміна етапів гри, етап виведення досягнень, підрахунок ігрових балів, вивід переможця.

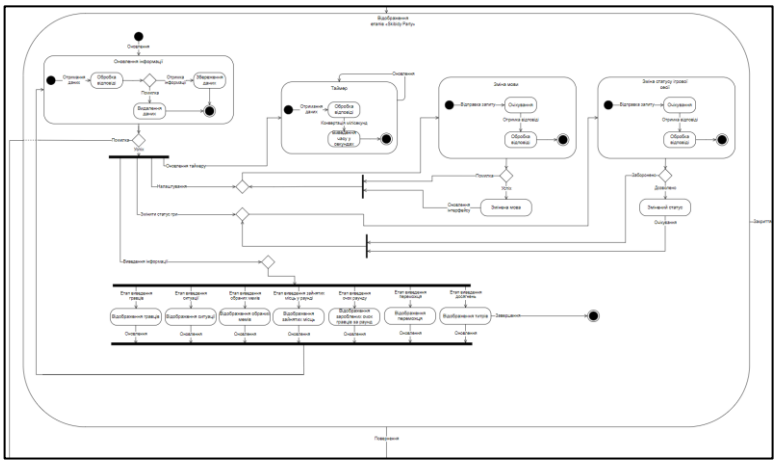


Рисунок 3.11 – UML діаграма станів екрану ігрового оточення, міні-гри «Skibidy Party», на клієнтській частині ігрового програмного застосунку (діаграма виконана самостійно)

Третя частина UML діаграми описує обробку логіки гри «Skibidy Party», а саме: відображення таймеру, збереження та видалення ігрових даних, зміна мови та гучності звуку, відображення титрів, відображення гравців, зміна ігрових ролей, відображення обраних мемів, додавання своїх мемів, механіка паузи, відображення поточного раунду.

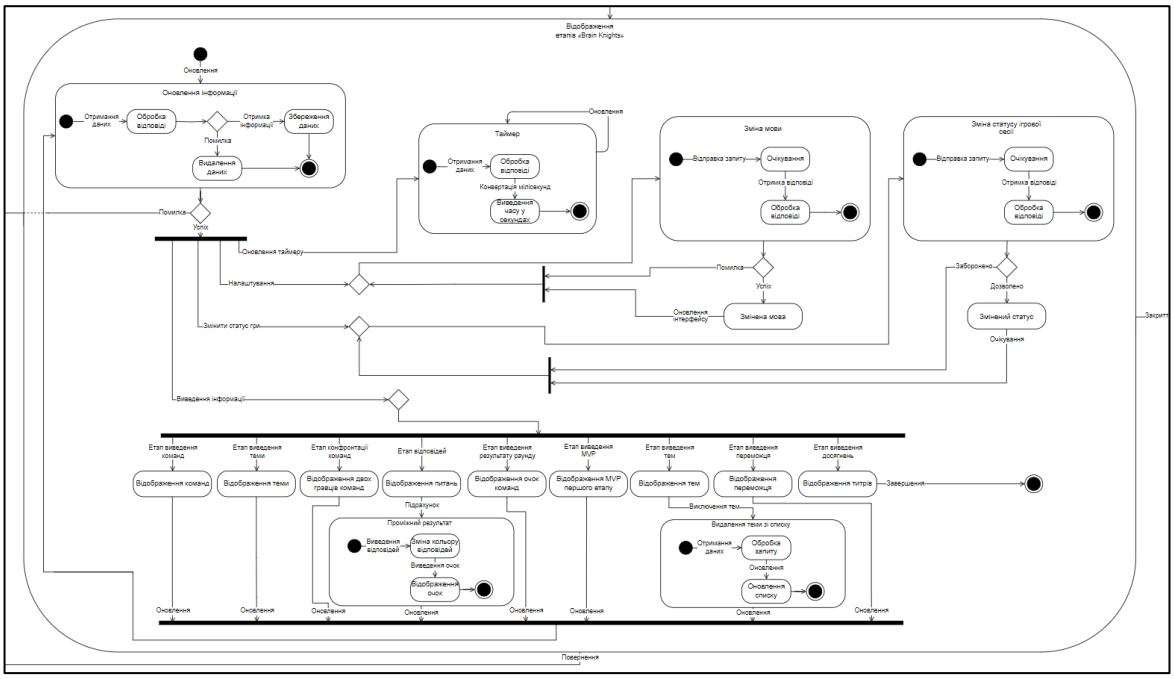


Рисунок 3.12 – UML діаграма станів core геймплею, міні-гри «Brain Knights», на клієнтській частині ігрового програмного застосунку (діаграма виконана самостійно)

Четверта частина UML діаграми описує обробку логіки гри «Brain Knights», а саме: відображення команд та гравців, відображення поточного раунду, зміна мови та гучності звуку, відображення титрів, відображення питань та тем, відображення найефективнішого гравця раунду, відображення переможців раунду та гри.

3.2 Проєктування архітектури ПЗ

Клієнтська частина програмної системи використовує класичну односторінкову архітектуру (SPA), що розділена на три рівні для ефективної організації функціональності. На першому рівні, що відповідає за відображення та взаємодію з користувачем, використовується бібліотека React для динамічного оновлення контенту. Другий рівень, який керує станом додатку та зберігає дані, базується на бібліотеці Redux [6], що забезпечує централізоване управління станом. На третьому рівні здійснюється взаємодія з сервером через REST API для ефективного обміну даними.

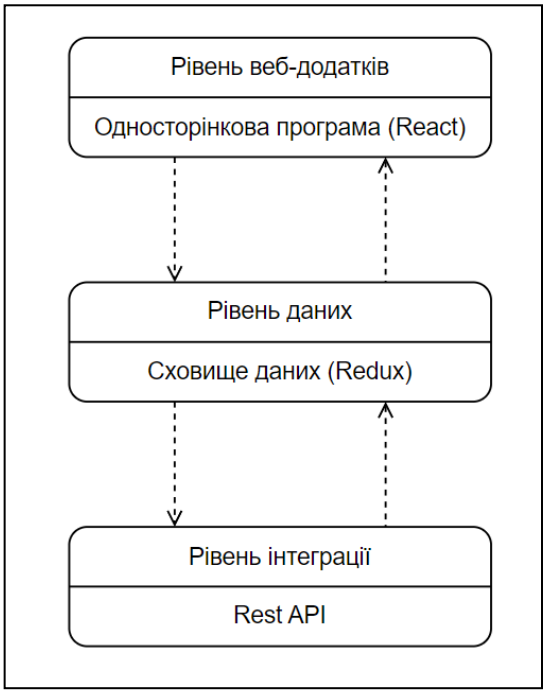


Рисунок 3.13 – Архітектура клієнтської частини проєкту (знімок екрана виконано самостійно)

Ця архітектура дозволяє розділити функціональність на логічно пов’язані рівні, спрощуючи розробку та підтримку. Кожен рівень має свої обов’язки та взаємодіє з іншими для повного функціонування додатку.

Для клієнтської частини системи було спроєктовано наступну структуру (див. рис. 4.18):

– «animations» для анімацій початку та завершення гри;

– «assets» для статичних ресурсів, таких як зображення, шрифти та музика;

– «components» для файлів, використовуваних на різних сторінках та модулях додатку;

– «layout» для файлів, що відповідають за загальний вигляд та розташування елементів на сторінці;

– «modules» для файлів різних модулів, використовуваних у додатку;

– «pages» для файлів, що відтворюють окремі сторінки додатку;

– «routes» для файлів, які визначають маршрутизацію;

– «services» для файлів, що оброблюють логіку взаємодії з сервером та локалізації;

– «store» для файлів, що відповідають за управління станом даних;

– «styles» для глобальних стилів;

– «utils» для допоміжних файлів та функцій.

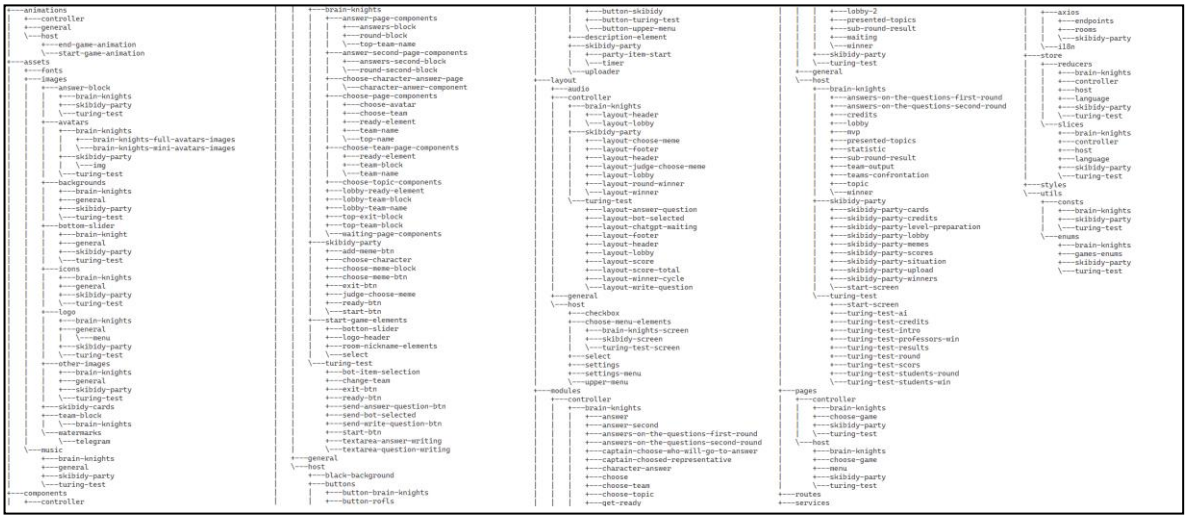


Рисунок 3.14 – Структура клієнтської частини проєкту (знімок екрана виконано самостійно)

Ця організація проєкту сприяє його зручній модернізації та підтримці у майбутньому.

3.3 Приклади найцікавіших алгоритмів та методів

Для досягнення мінімальної затримки використовується розподілення обчислювальної логіки на 3 окремих підмодулі: додаток на смартфоні збирає вихідну інформацію та передає її на сервер, де в свою чергу підраховується основна ігрова логіка, після обробки він передає дані до клієнтської частини гри де відображається ігрове оточення, браузерний клієнт в такій системі бере на себе навантаження щодо прорахунку всього графічного оточення [7].

На сервері використовуються методи updateGameState(newState): одночасно оновлює стан гри та синхронізує його з усіма клієнтами. Та optimizeServerLoad(): Здійснює оптимізацію роботи сервера для забезпечення мінімальної затримки та ефективного використання ресурсів шляхом видалення тимчасових файлів.

На екрані ігрового оточення у клієнтській частині використовується модульна система сторінок, та система зміни стану useState(), за допомогою якої система змінює HTML код односторінкової гри без перезавантаження сторінки.

Веб-застосунок для смартфону реалізований за допомогою бібліотеки React для розробки інтерфейсу контролера, та методів React Hooks, оскільки вони є більш оптимізованими з точки зору продуктивності, ніж альтернативний підхід з використанням методів Redux, що важливо для мінімізації затримки відгуку від смартфона [8].

На контролері метод reducer(data): Передає зібрані дані на сервер для обробки та синхронізації з іншими гравцями, залежності від етапу гри який змінюється за заданим часом. Дані передаються за допомогою технології постійного підключення WebSocket.

Затримка відгуку в такій системі варіюватиметься від 5 до 40 мілісекунд залежно від навантаження на сервер. Ця система добре підходить в іграх із малозначущою ігровою затримкою – вікторини, головоломки, паті Модульність дає можливість легко масштабувати систему і змінювати її при потребі.

3.4 Створення UI / UX або іншого дизайну системи

Під час проєктування інтерфейсу та левел-дизайну, був розроблений UI макет ігрових екранів, та в подальшому реалізований в ігровому застосунку [9-10].

На рисунку 3.15 зображений інтерфейс стартового екрану, через який користувач потрапляє до меню вибору гри на головному екрані оточення.

Рисунок 3.15 – Інтерфейс стартового екрану гри (знімок екрана виконано самостійно)

На рисунку 3.16 зображений інтерфейс стартового екрану мобільного контролеру, через який користувач потрапляє до ігрової кімнати. За допомогою полів для введення, гравець може додати код кімнати та своє ім’я.

Рисунок 3.16 – Інтерфейс стартового екрану гри на мобільному контролері (знімок екрана виконано самостійно)

На рисунку 3.17 зображений інтерфейс меню вибору гри, для навігації на екрані використовується модуль слайдеру, якщо натиснути на кнопку «Меню», користувач повернеться до стартового екрану, якщо натисне на кнопку «Налаштування», користувач перейде до налаштувань гри, де можна змінити локалізації системи, та гучність звуку в грі.

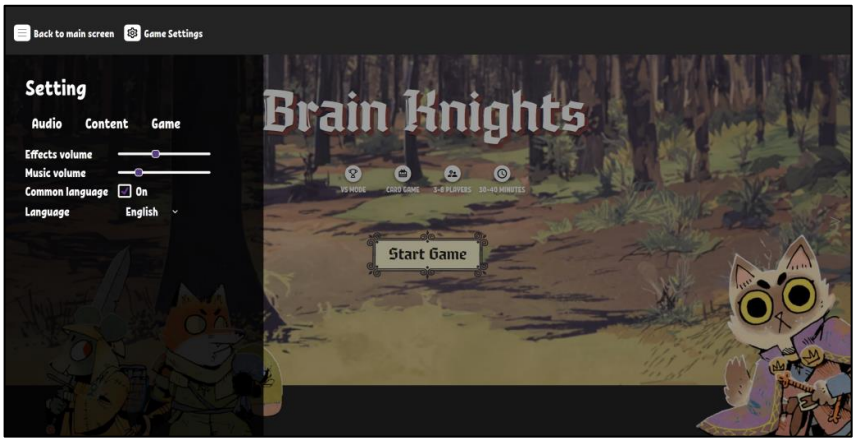


Рисунок 3.17 – Інтерфейс меню вибору гри (знімок екрана виконано самостійно)

На рисунку 3.18 зображена ігрова кімната, де відображені гравці, які підключені до нього, також відображений ігровий код, за яким гравці можуть під’єднатись до нього. За допомогою контролеру гравці можуть підтвердити готовність, та почати гру. Еквіваленти зображеного лобі розроблені для всіх ігрових режимів.

Рисунок 3.18 – Інтерфейс ігрової кімнати на головному екрані гри в грі «Brain Knights» (знімок екрана виконано самостійно)

Інтерфейси гри «Brain Knights» для інтерфейсу екрану оточення. На рисунку 3.19 після початку гри показується тема, яку будуть розігрувати гравці в раунді.

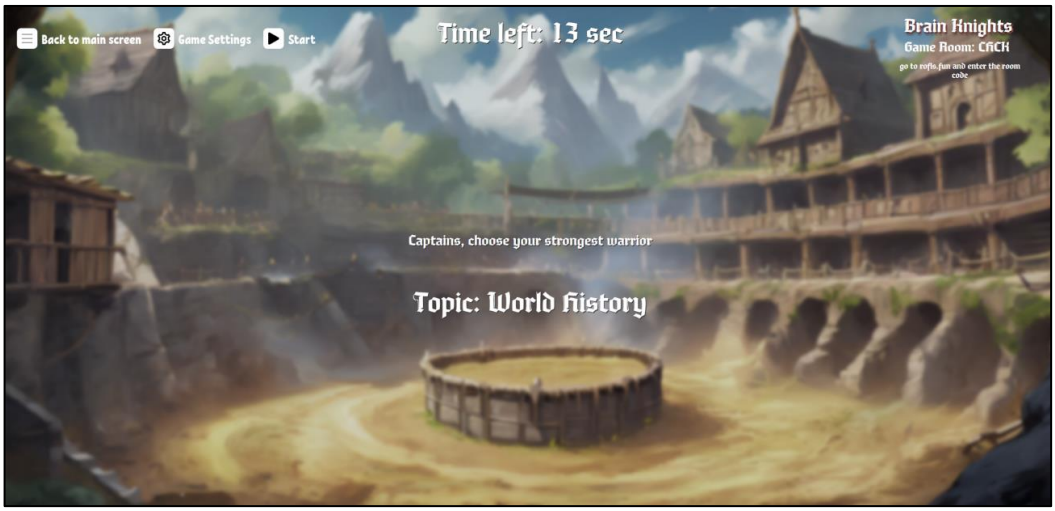


Рисунок 3.19 – Інтерфейс теми раунду (знімок екрана виконано самостійно)

На рисунку 3.20 зображено гравців які будуть відповідати на питання у бліц-раунді.

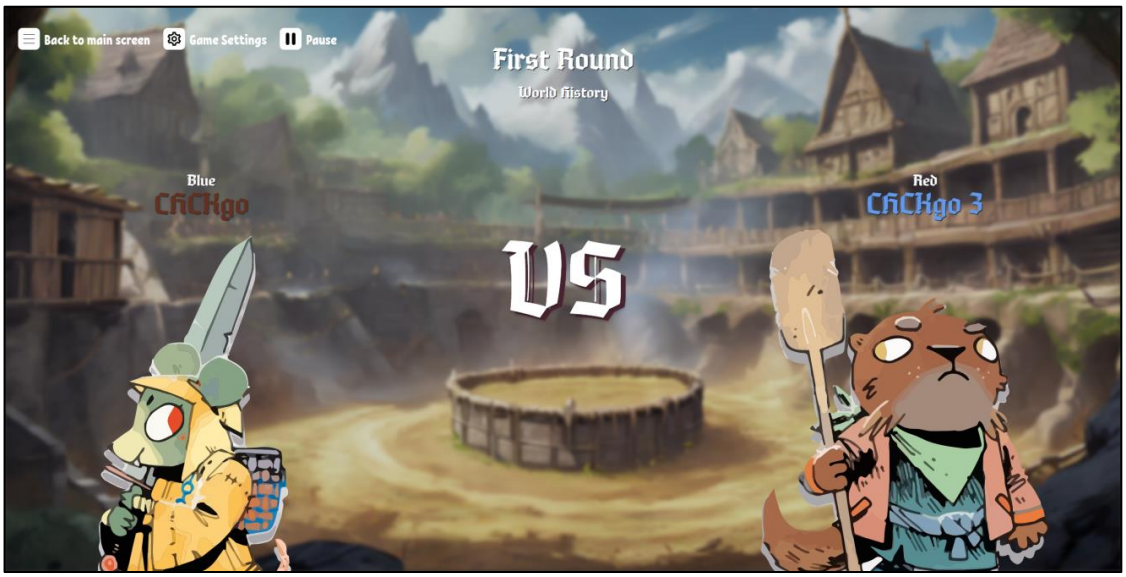


Рисунок 3.20 – Інтерфейс початку бліц-раунду (знімок екрана виконано самостійно)

На рисунку 3.21 зображений бліц-раунд в якому гравці відповідають на швидкість, в центрі екрана зображене запитання та можливі відповіді на нього, по краям екрану зображений рахунок гравців.

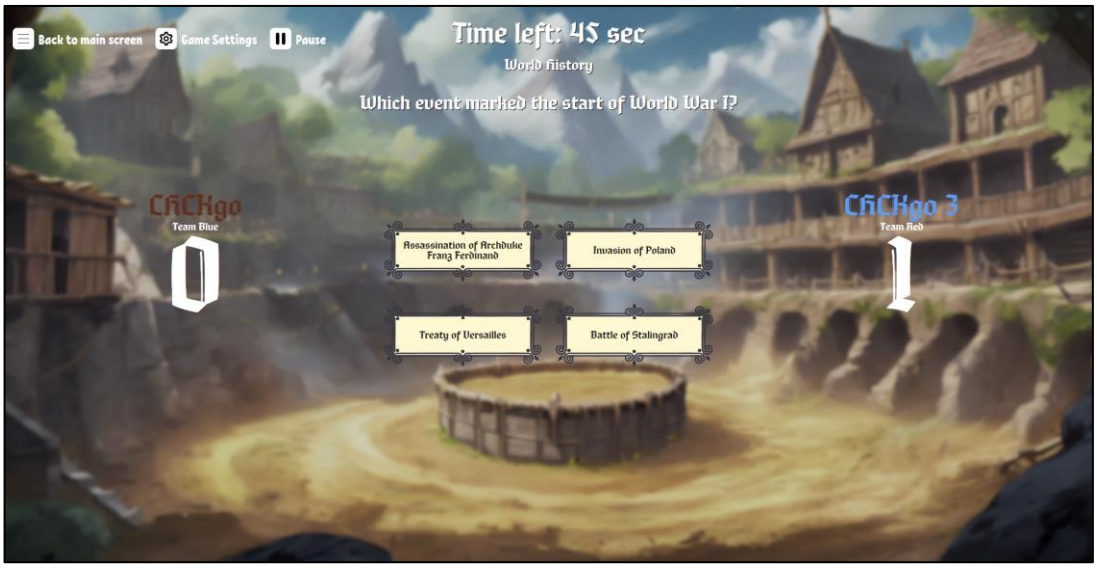


Рисунок 3.21 – Інтерфейс ігрового раунду (знімок екрана виконано самостійно)

На рисунку 3.22 зображений гравець який набрав більшу кількість балів у блітз-раунді.

Рисунок 3.22 –Інтерфейс переможця блітз-раунду (знімок екрана виконано самостійно)

На рисунку 3.23 зображені загальний рахунок після ігрового раунду. По центру екрану зображений ігровий рахунок, по бокам назви команд та аватари гравців.

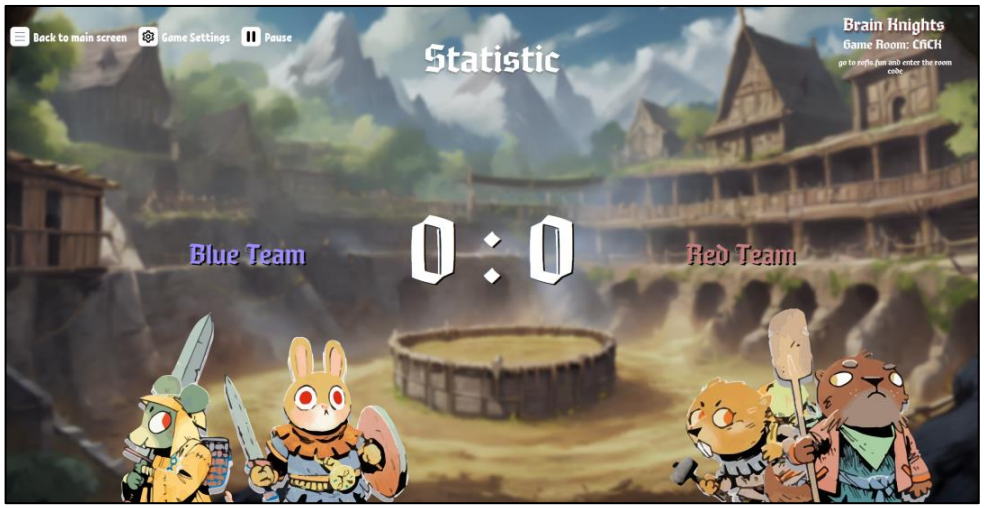


Рисунок 3.23 –Проміжні результати раунду (знімок екрана виконано самостійно)

Після того як всі гравці зіграли по одному раунді у бліц-опитуваннях, гра переходить до другої стадії, де гравцям потрібно командно відповідати на складні питання. На рисунку 3.24 зображені теми які капітани по черзі видаляють зі списку за допомогою мобільного контролера.

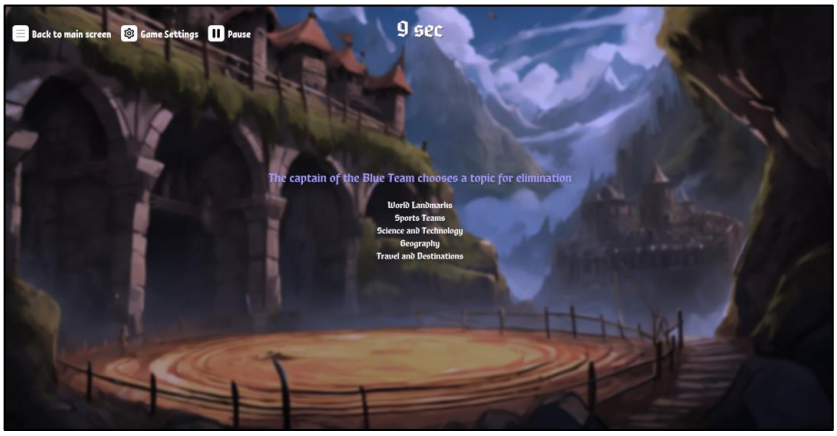


Рисунок 3.24 – Вибір теми для другої стадії гри «Brain Knights» (знімок екрана виконано самостійно)

На рисунку 3.25 зображений ігровий раунд у другій стадії, де гравцям потрібно командно відповідати на складні питання. Над обраною командою відповіддю, зображуються аватари гравців. По бокам екрану показується назва команди та загальний рахунок.

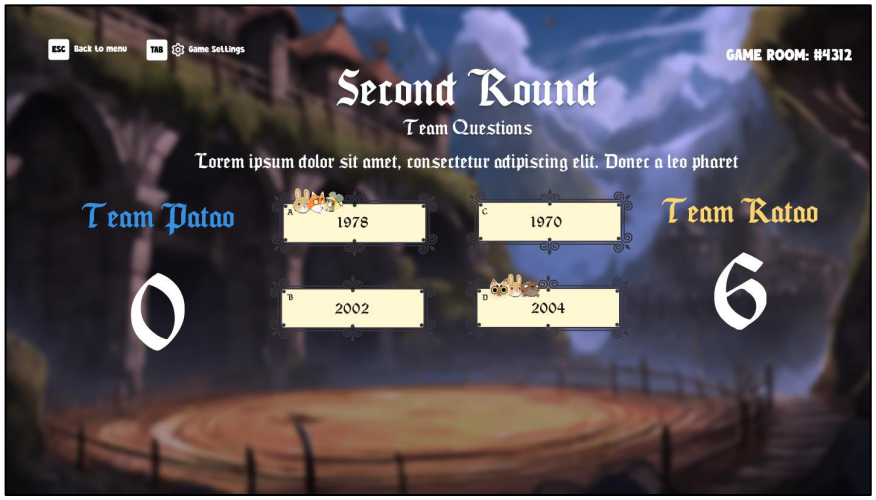


Рисунок 3.25 – Ігровий раунд у другій стадії гри «Brain Knights» (знімок екрана виконано самостійно)

На рисунку 3.26 зображений вибір ігрового аватару при вході до ігрової кімнати на мобільному ігровому контролері.

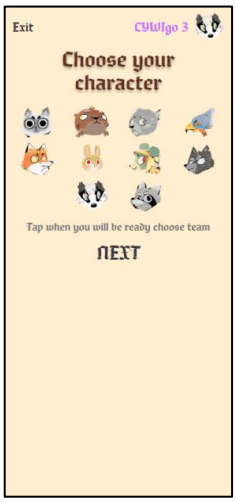


Рисунок 3.26 – Вибір ігрового аватара на мобільному контролері в грі «Brain Knights» (знімок екрана виконано самостійно)

На рисунку 3.27 зображений вибір назви команди яку може зробити капітан команди. Також гравець може змінити команду та стати капітаном при необхідності, після того як гравці натиснуть на кнопку «Ready» гра почнеться.

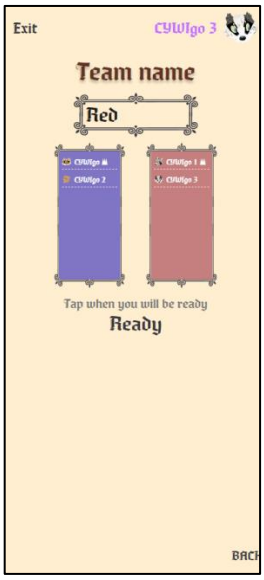


Рисунок 3.27 – Вибір команди на мобільному ігровому контролері в грі «Brain Knights» (знімок екрана виконано самостійно)

На рисунку 3.28 зображений вибір гравця для участі у блітз-опитуванні, який капітан може зробити. Після вибору гравця один раз, він буде заблокований, а аватар в свою чергу змінить свій колір на сірий.

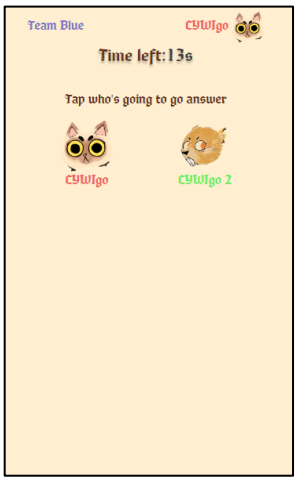


Рисунок 3.28 – Вибір гравця для відповіді на мобільному ігровому контролері в грі «Brain Knights» (знімок екрана виконано самостійно)

На рисунку 3.29 зображений вибір відповіді гравцем на запитання, на екрані також зроблений таймер, який показує залишок часу до кінця раунду.

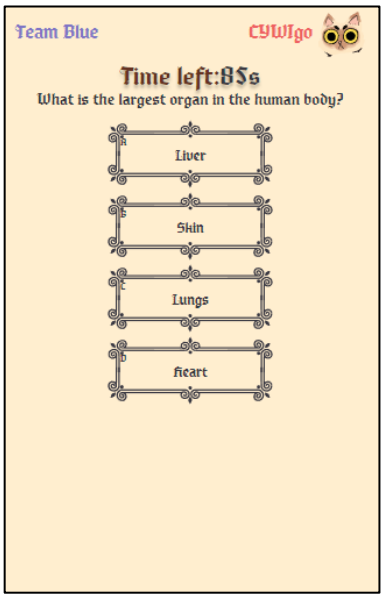


Рисунок 3.29 – Вибір відповіді на запитання в грі «Brain Knights» (знімок екрана виконано самостійно)

На рисунку 3.30 зображена ігрова кімната гри «Skibidy Party», де відображені гравці, які підключені до нього, також відображений ігровий код, за яким гравці можуть під’єднатись до нього.

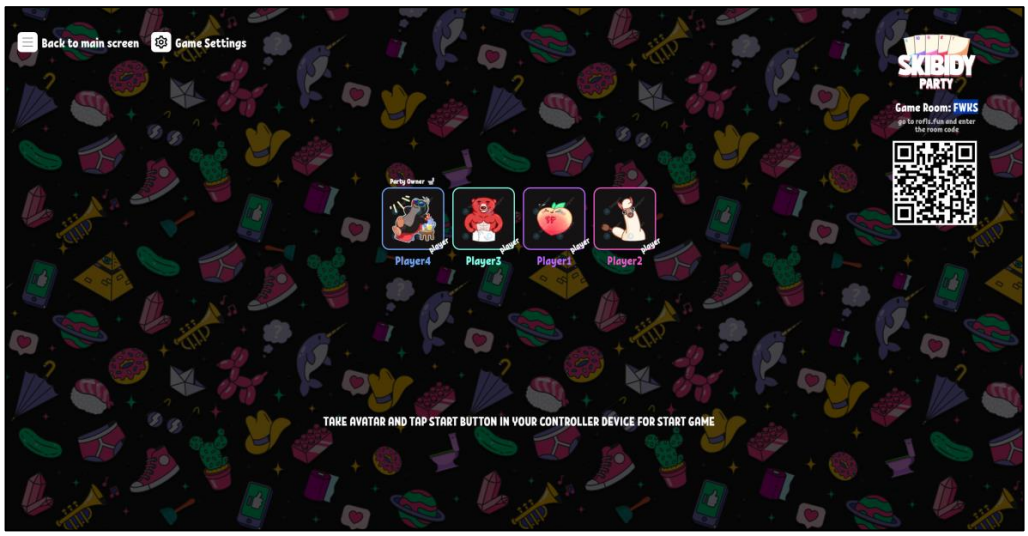


Рисунок 3.30 – Інтерфейс ігрової сесії в грі «Skibidy Party» (знімок екрана виконано самостійно)

На рисунку 3.31 зображене ігрове поле, на якому відбувається розподіл карт мемів та ситуацій перед початком раунду.

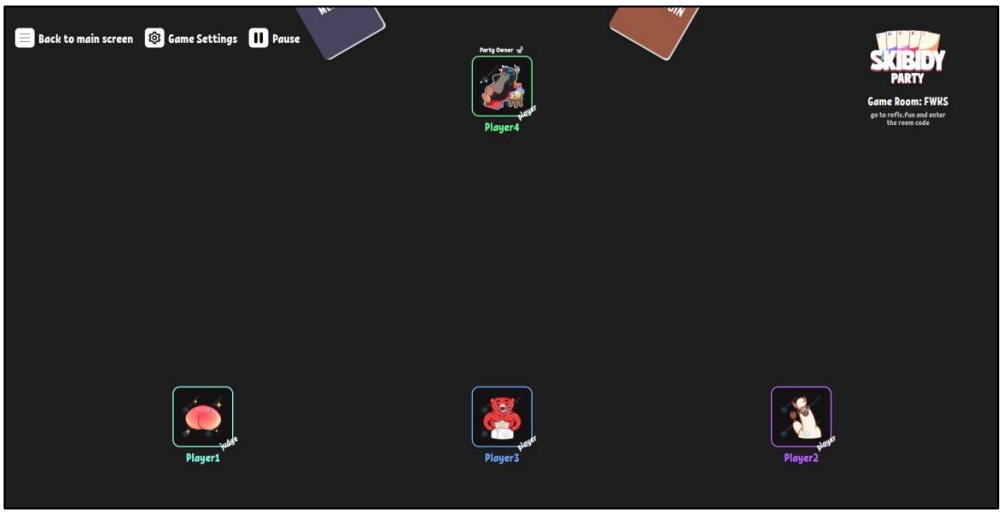


Рисунок 3.31 – Інтерфейс ігрового поля в «Skibidy Party» (знімок екрана виконано самостійно)

На рисунку 3.32 зображений інтерфейс ігрової ситуація, до якої гравцям потрібно буде обрати мем з набору гравця.

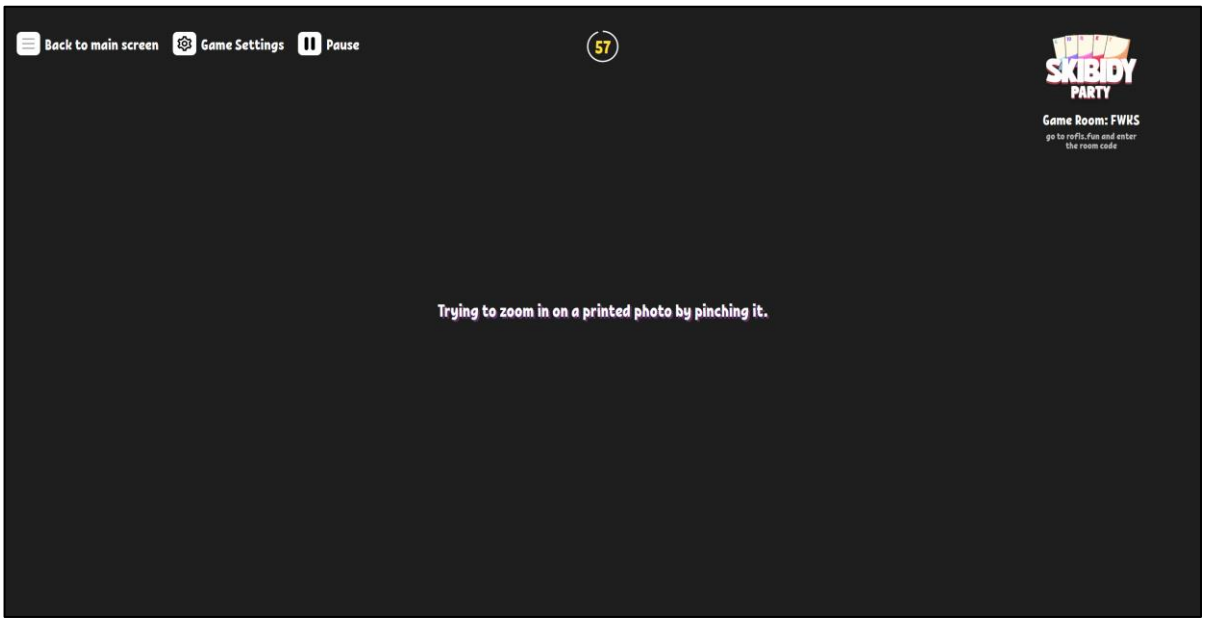


Рисунок 3.32 – Інтерфейс ігрової ситуації в « Skibidy Party» (знімок екрана виконано самостійно)

На рисунку 3.33 зображений інтерфейс з обраними гравцями мемами, гравець який має роль судді має можливість оцінити їх від 1 до 3 місця.

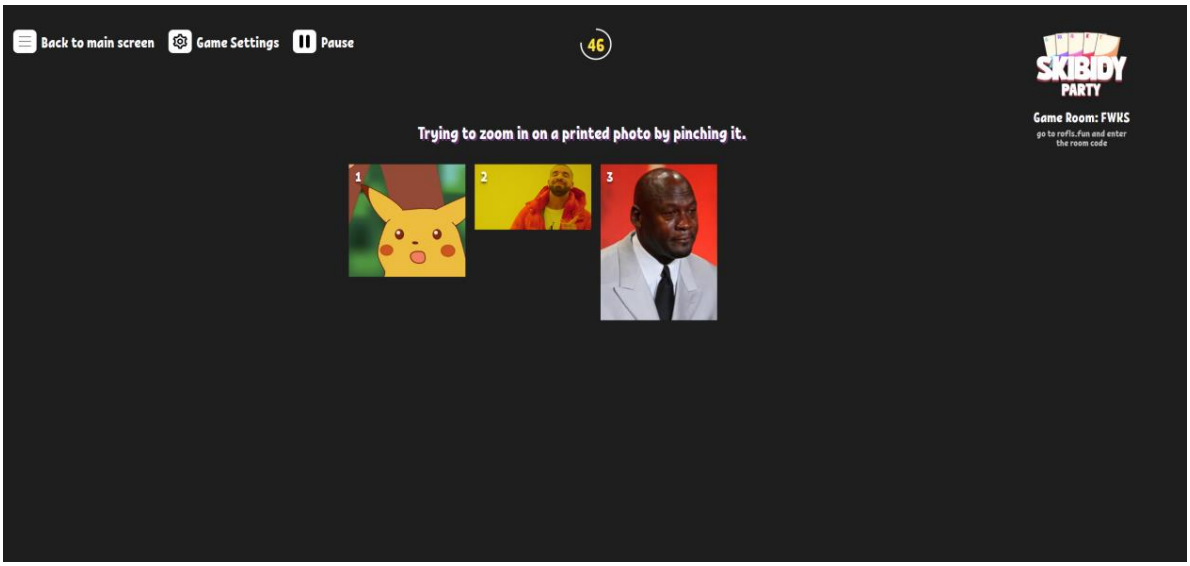


Рисунок 3.33 – Інтерфейс обраних мемів в « Skibidy Party» (знімок екрана виконано самостійно)

На рисунку 3.34 зображений інтерфейс переможців раунду, по центру відображається користувач який зайняв перше місце, зліва від нього гравець який зайняв друге, справа гравець який зайняв третє місце.

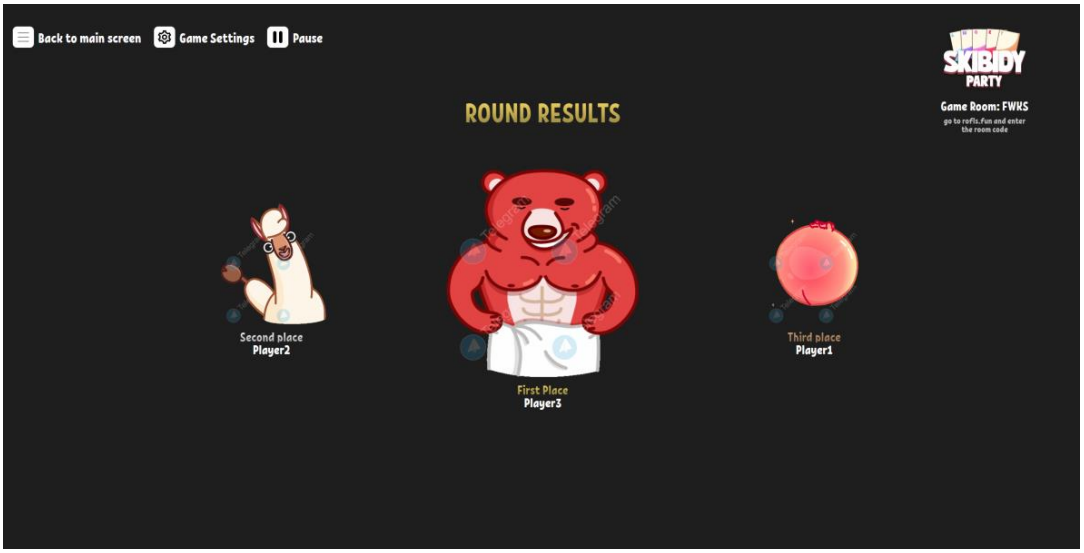


Рисунок 3.34 – Інтерфейс переможців раунду в « Skibidy Party» (знімок екрана виконано самостійно)

На рисунку 3.35 зображена статистика після раунду. Після кожного раунду статистика динамічно оновлюється, і аватари гравців міняються місцями.

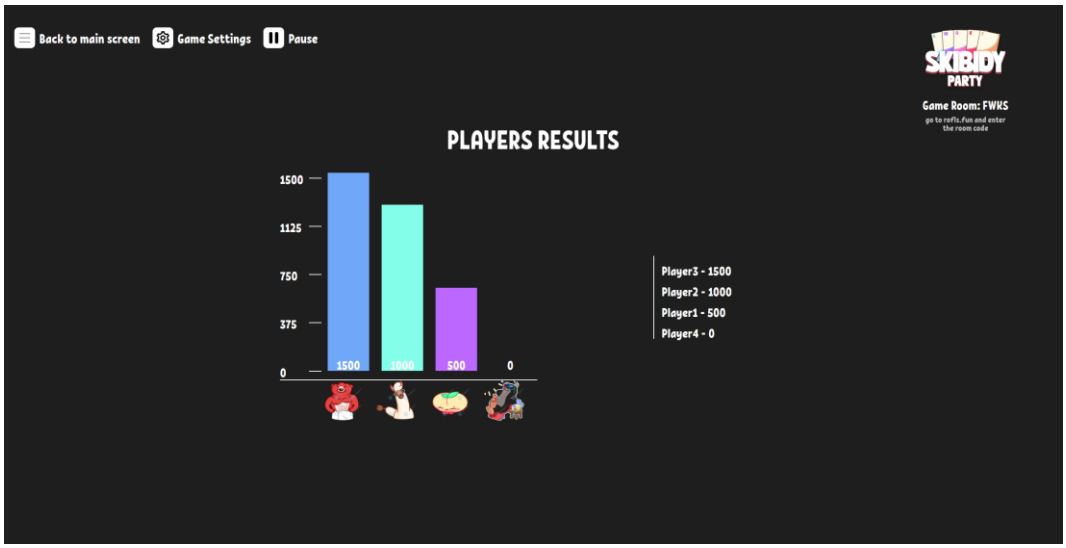


Рисунок 3.35 – Статистика гри в « Skibidy Party» (знімок екрана виконано самостійно)

На рисунку 3.36 зображений інтерфейс вибору аватару на мобільному контролері, під аватарами є дві кнопки: «Ready» відповідає за зміну статусу гравця для початку гри, «Exit» відповідає за вихід з гри та повернення на головний екран.

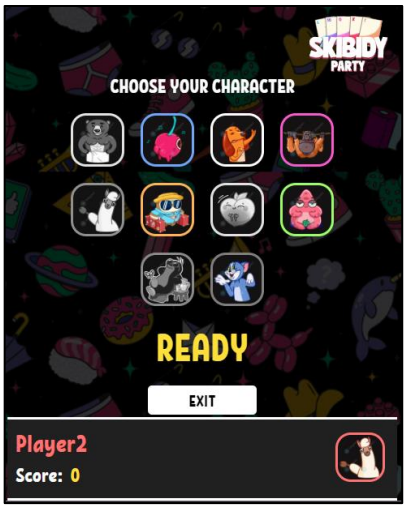


Рисунок 3.36 – Вибір аватару в «Skibidy Party» на мобільному контролері (знімок екрана виконано самостійно)

На рисунку 3.37 зображений інтерфейс вибору мему, для цього гравцю потрібно натиснути на картинку з мемом, та натиснути кнопку «Add meme».

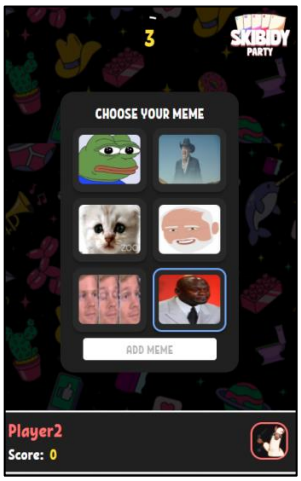


Рисунок 3.37 – Вибір мему в «Skibidy Party» на мобільному контролері (знімок екрана виконано самостійно)

На рисунку 3.38 зображений інтерфейс вибору переможного мему суддею, для цього гравцю потрібно натиснути на картинку з мемом за чергою, якщо гравець хоче змінити обрані місця, він може ще раз натиснути на меми в новому порядку. Після цього гравцю потрібно натиснути кнопку «Confirm».

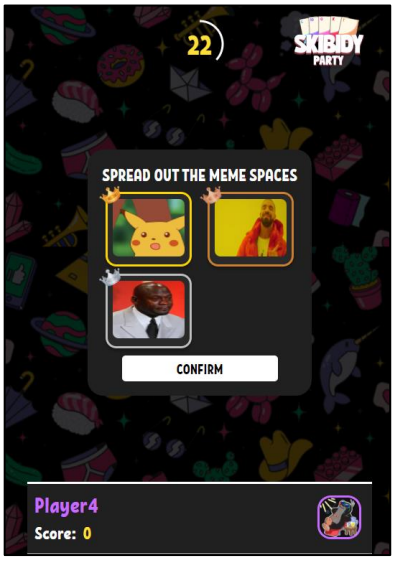


Рисунок 3.38 – Вибір переможного мему в «Skibidy Party» на мобільному контролері (знімок екрана виконано самостійно)

На рисунку 3.39 зображена ігрова кімната гри «Turing Test», де відображені гравці, які підключені до нього, також відображений ігровий код, за яким гравці можуть під’єднатися до нього.

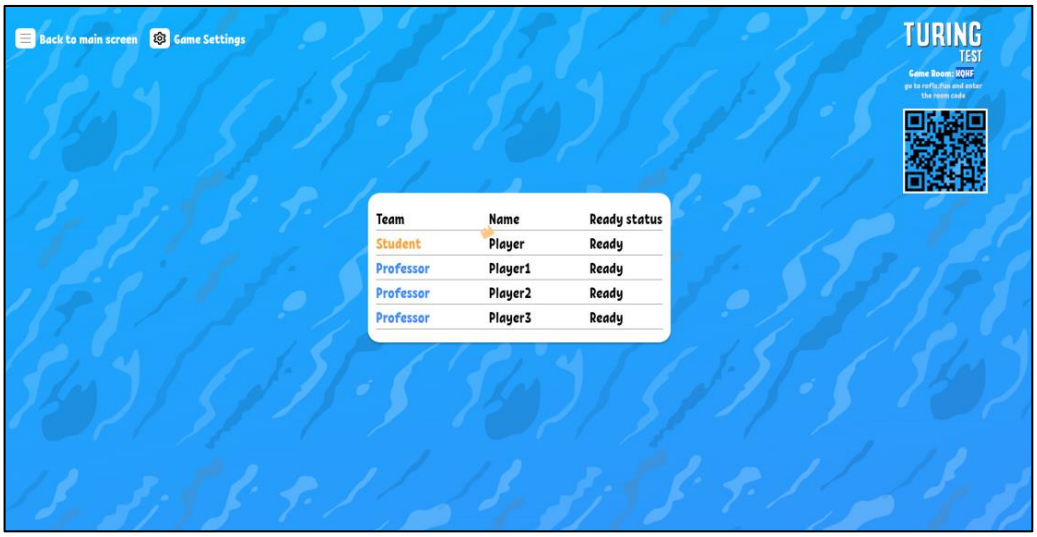


Рисунок 3.39 – Інтерфейс ігрової сесії в грі «Turing Test» (знімок екрана виконано самостійно)

На рисунку 3.40 зображений екран раунду, коли гравці з роллю студента на своїх контролерах вписують питання для професорів та ШІ.

Рисунок 3.40 – Інтерфейс раунду студента в грі «Turing Test» (знімок екрана виконано самостійно)

На рисунку 3.41 зображений екран раунду, коли гравці з роллю професора на своїх контролерах вписують відповіді на питання студентів.

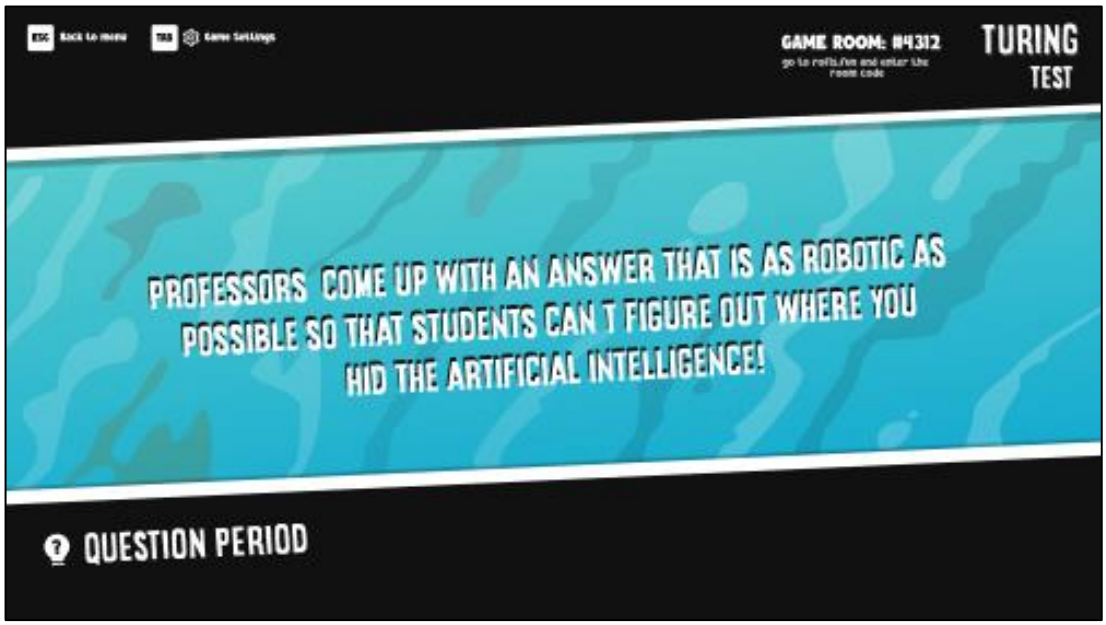


Рисунок 3.41 – Інтерфейс раунду професора в грі «Turing Test» (знімок екрана виконано самостійно)

На рисунку 3.42 зображений інтерфейс пошуку та вибору відповіді ШІ студентами.

Рисунок 3.42 – Інтерфейс пошуку відповіді ШІ в грі «Turing Test» (знімок екрана виконано самостійно)

На рисунку 3.43 зображений інтерфейс після вибору відповіді, якщо відповідь правильна, вона стає синьою, якщо неправильна – червоною.

Рисунок 3.43 – Інтерфейс після пошуку відповіді ШІ в грі «Turing Test» (знімок екрана виконано самостійно)

На рисунку 3.44 зображений інтерфейс переможця раунду, в цьому випадку це студент.

Рисунок 3.44 – Інтерфейс переможця якщо це студент в грі «Turing Test» (знімок екрана виконано самостійно)

На рисунку 3.45 зображений інтерфейс переможця раунду, в цьому випадку це професор.

Рисунок 3.45 – Інтерфейс переможця якщо це професор в грі «Turing Test» (знімок екрана виконано самостійно)

На рисунку 3.46 зображений інтерфейс рахунку кожної ітерації раундів та в кінці гри.

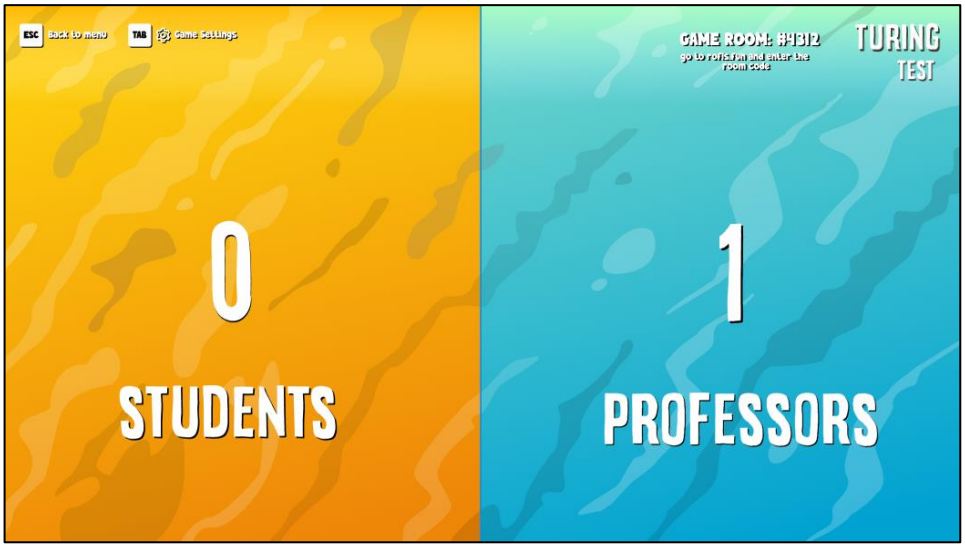


Рисунок 3.46 – Інтерфейс рахунку в грі «Turing Test» (знімок екрана виконано самостійно)

На рисунку 3.47 зображений інтерфейс вибору команди на мобільному контролері. На екрані є три кнопки: «Ready» відповідає за зміну статусу гравця для початку гри, «Exit» відповідає за вихід з гри та повернення на головний екран, та «Start» для початку гри якщо користувач є адміністратором кімнати.

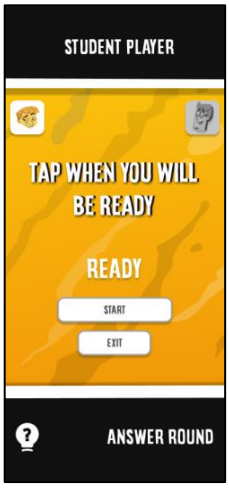


Рисунок 3.47 – Інтерфейс вибору команди на мобільному контролері в грі «Turing Test» (знімок екрана виконано самостійно)

На рисунку 3.48 зображений інтерфейс вписування питання студентом, для відправки питання треба натиснути на кнопку «Send».

Рисунок 3.48 – Інтерфейс вписування питання командою студентів на мобільному контролері в грі «Turing Test» (знімок екрана виконано самостійно)

На рисунку 3.49 зображений інтерфейс вписування відповіді на питання командою професорів, для відправки питання треба натиснути на кнопку «Send».

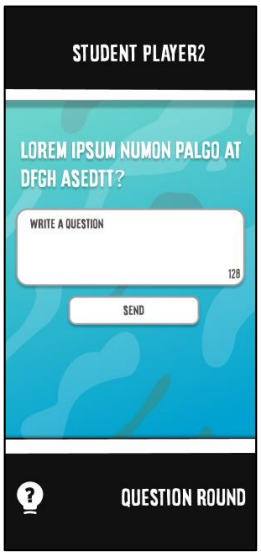


Рисунок 3.49 –Інтерфейс додавання відповіді на питання командою професорів на мобільному контролері в грі «Turing Test» (знімок екрана виконано самостійно)

На рисунку 3.50 зображений інтерфейс вибору відповіді яка на думку студентів є варіантом від ШІ, після вибору номера відповіді для остаточного голосування потрібно натиснути на кнопку «Send».

Рисунок 3.50 – Інтерфейс вибору відповіді ШІ на мобільному контролері в грі «Turing Test» (знімок екрана виконано самостійно)

На рисунку 3.51 зображений інтерфейс рахунку кожної ітерації раундів та в кінці гри на мобільному контролері в грі «Turing Test».

Рисунок 3.51 – Інтерфейс рахунку на мобільному контролері в грі «Turing Test» (знімок екрана виконано самостійно)

Всі інтерфейси розроблені з дотримкою вимог зручність використання, простоти. Кожна гра має свій стить який відображає її унікальність та підкреслює геймплейні відмінності.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

Клієнтська частин ігрового програмного застосунку розроблена за допомогою бібліотеки React з використанням мови програмування JavaScript. Для забезпечення обміну даними в реальному часі з сервером була використана бібліотека Socket.IO [10] разом з інструментами WebSocket. Список бібліотек та їх версій, які були використані в клієнтській частині наведено в таблиці 4.1.

Таблиця 4.1 – Список бібліотек та їх версій, які використовуються в back-end частині ігрового програмного застосунку

Назва	Версія	Призначення
axios	1.6.2	Використовується для виконання HTTP-запитів з Node.js
framer-motion	10.16.14	Використовується для створення комплексних анімацій
redux	4.2.1	Використовується для керування станом елементів у застосунку
sass	1.0.1	Розширення мови CSS, що дозволяє використовувати змінні, вкладені правила, міксини, функції та інші корисні інструменти для полегшення написання стилів.
i18n-iso-countries	7.2.0	Використовується для створення багатоязичного інтерфейсу програмного застсоунку
i18next-browser-languagedetector	7.2.1	
i18next-http-backend	2.4.1	

Кінець таблиці 4.1

Назва	Версія	Призначення
uuid	9.0.1	Використовується для створення унікальних ідентифікаторів.
joi	17.11.0	Бібліотека для валідації даних
react-router-dom	6.19.0	Проміжне програмне забезпечення для обробки файлових завантажень у Node.js
socket.io	4.7.5	Бібліотека для роботи з WebSocket у Node.js.

Для налаштування порта клієнтської частини та зв’язку з серверною частиною, у файлі змінних середовища було створено дві змінні: PORT, що вказує порт, на якому буде запущено клієнтський застосунок; а також REACT_APP_BACKEND_URL, що вказує на адресу, за якою працює сервер.

Для відправлення запитів налаштовано проксі-сервер додатка, який перенаправляє всі запити, що починаються з /api, на вказаний цільовий сервер:

```
module.exports = function (app) {
    app.use(
        "/api",
        createProxyMiddleware({
            target: target,
            changeOrigin: true,
        })
    );
};
```

Для налаштування аватарів гравців у міні-іграх Brain Knights та Skibidy Party було створено два типи зображень: масив об’єктів, що містить назву аватара, зображення та колір:

```
const brainKnightsMiniAvatarsArray = [
  { avatar: "cat", image: miniCat, color: "#EF6161" },
  { avatar: "bobak", image: miniBobak, color: "#619AEF" },
  { avatar: "beaver", image: miniBeaver, color: "#63EF61" },
  { avatar: "eagle", image: miniEagle, color: "#499A3C" },
  { avatar: "fox", image: miniFox, color: "#EC61EF" },
  { avatar: "hare", image: miniHare, color: "#383741" },
  { avatar: "mouse", image: miniMouse, color: "#643121" },
  { avatar: "wolf", image: miniWolf, color: "#DE9300" },
```

```
  { avatar: "badger", image: miniBadger, color: "#C76CFF" },
  { avatar: "raccoon", image: miniRaccoon, color: "#808080" },
];
```

Для управління станом хоста у додатку було створено та налаштовано зріз стану з використанням бібліотеки Redux Toolkit. Код визначає початковий стан, який включає об’єкт хоста з полем id, що спочатку дорівнює null, і об’єкт lastSelectedGame з полями id та name, також рівними null. Зріз hostSlice містить три редюсери: initializeId, setLastSelectedGame та resetLastSelectedGame, які викликають відповідні функції з файлу host.reducers:

```
const initialState = {
  host: {
    id: null,
  },
  lastSelectedGame: {
    id: null,
    name: null,
  },
};
const hostSlice = createSlice({
  name: "host",
  initialState,
  reducers: {
    initializeId: (state) => hostReducers.initializeId(state),
    setLastSelectedGame: (state, action) =>
      hostReducers.setLastSelectedGame(state, action),
    resetLastSelectedGame: (state, initialState) =>
      hostReducers.resetLastSelectedGame(state, initialState),
  },
});
```

Зріз стану потрібен для управління ідентифікатором хоста та останньою вибраною грою в додатку. Функція initializeId перевіряє наявність ідентифікатора хоста в куках. Якщо ідентифікатор існує, він встановлюється в стан, якщо ні, створюється новий ідентифікатор з допомогою uuidv4 та зберігається в куках. Функція setLastSelectedGame оновлює стан останньої вибраної гри на основі переданого дії, resetLastSelectedGame повертає стан останньої вибраної гри до значень початкового стану:

```
const initializeId = (state) => {
    const currentHostId = Cookies.get("hostId");
    if (currentHostId) {
        state.host.id = currentHostId;
    } else {
        const id = uuidv4();
        state.host.id = id;
        Cookies.set("hostId", id);
    }
```

```
};
const setLastSelectedGame = (state, action) => {
    state.lastSelectedGame = action.payload;
};
const resetLastSelectedGame = (state, initialState) => {
    state.lastSelectedGame = initialState.lastSelectedGame;
};
```

Для управління мовними налаштуваннями в додатку також було створено та налаштовано зріз стану з використанням бібліотеки Redux Toolkit. Код визначає початковий стан, який включає мовні параметри: language, personalControllerLanguage, personalHostLanguage, всі з яких за замовчуванням встановлені на «en», та прапорець commonLanguage, що також за замовчуванням встановлений на true. Зріз languageSlice містить п’ять редюсерів: initialize, setLanguage, setPersonalControllerLanguage, setPersonalHostLanguage та setCommonLanguage, які викликають відповідні функції з файлу language.reducers:

```
const initialState = {
    language: 'en',
    personalControllerLanguage: 'en',
    personalHostLanguage: 'en',
    commonLanguage: true,
}
const languageSlice = createSlice({
    name: 'language',
    initialState,
    reducers: {
        initialize: state => languageReducers.initialize(state),
        setLanguage:           (state,           action)           =>
languageReducers.setLanguage(state, action),
        setPersonalControllerLanguage: (state, action) =>
            languageReducers.setPersonalControllerLanguage(state,
action),
        setPersonalHostLanguage: (state, action) =>
            languageReducers.setPersonalHostLanguage(state, action),
        setCommonLanguage: (state, action) =>
            languageReducers.setCommonLanguage(state, action),
    },
})
```

Функція initialize перевіряє наявність мовних налаштувань у cookie, якщо вони порожні або містять невалідні дані, вона встановлюється значення «en» за замовчуванням. Це ж саме стосується мов для контролера та хоста. Функція setLanguage оновлює основну мову додатка, setCommonLanguage оновлює мову на всіх підключенних пристроях до ігрової кімнати, setPersonalControllerLanguage оновлює мову для контролера, setPersonalHostLanguage оновлює мову для хоста. Всі оновлені дані, що зберігаютсья у Redux, також зберігаються у куках файлах.

Для керування відтворенням музики та звукових ефектів на вебсторінці було створено змінні та функції, що дозволяють перемикати треки, відтворювати звуки та змінювати джерела аудіо. Змінні music та audio відповідають за отримання елементів HTML з відповідними ідентифікаторами:

```
const music = document.getElementById('myMusic');
const audio = document.getElementById('myAudio');
```

Функція prevItem змінює поточний індекс на попередній у циклічному порядку, відтворює звук «whoosh» і через півсекунди змінює індекс ще раз. Через секунду після виклику функції перевіряє індекс і змінює джерело музики відповідно до індексу: 0 – «musicsbrainknights», 1 – «musicskibidy», 2 – «musicsturingtest». Знову відтворюється звук «whoosh2» та очищується таймаут:

```
const prevItem = () => {
    setCurrentIndexAnimation(
        prevIndex => (prevIndex - 1 + itemsLength) % itemsLength
    )
    playSound(whoosh)
    const timeoutId = setTimeout(() => {
        setCurrentIndex(prevIndex => (prevIndex - 1 + itemsLength) %
itemsLength)
    }, 500)
    const timeoutId2 = setTimeout(() => {
        if ((currentIndexAnimation - 1 + itemsLength) % itemsLength ===
0) {
            changeSource(musicsbrainknights)
        }
        if ((currentIndexAnimation - 1 + itemsLength) % itemsLength ===
1) {
            changeSource(musicsskibidy)
        }
        if ((currentIndexAnimation - 1 + itemsLength) % itemsLength ===
2) {
            changeSource(musicsturingtest)
        }
        playSound2(whoosh2)
    }, 1000)
    return () => clearTimeout(timeoutId, timeoutId2)
}
```

Функція playSound приймає нове джерело, змінює його для елемента audio, встановлює гучність і відтворює звук, playSound2 робить те саме, але для іншого елемента аудіо – audio2, changeSourceEffect змінює джерело для аудіо, зупиняє його, перезавантажує і викликає playSound:

```
const playSound = newSource => {
    if (newSource) {
        changeSourceEffect(newSource, audio)
    }
```

```
        if (audio) {
            audio.volume = volumeEffects
            audio.play()
        }
    }
    const playSound2 = newSource => {
        if (newSource) {
            changeSourceEffect(newSource, audio2)
        }
        if (audio2) {
            audio2.volume = volumeEffects
            audio2.play()
        }
    }
    const changeSourceEffect = (newSource, audio) => {
        audio.pause()
        audio.src = newSource
        audio.load()
        playSound()
    }
```

Функція playMusic відтворює музику з елемента music після встановлення гучності, changeSource змінює джерело для елемента music, зупиняє поточне відтворення, перезавантажує і викликає playMusic:

```
    const playMusic = () => {
        if (music) {
            music.volume = volumeMusic
            music.play()
        }
    }
    const changeSource = newSource => {
        music.pause()
        music.src = newSource
        music.load()
        playMusic()
    }
```

Для реалізації маршрутизації на клієнтській частині, яка призначена для хоста, використовується React бібліотека react-router-dom:

```
    import React from 'react'
    import { Routes, Route, Navigate } from 'react-router-dom'
    import {
      HostBrainKnights,
      HostChooseGame,
      HostTuringTest,
      HostSkibidyParty,
    } from '../pages'
    export const useRoutes = () => {
      return (
        <Routes>
          <Route path='/host/menu' exact element={<HostChooseGame />} />
          <Route              path='/host/brainknights'              exact
element={<HostBrainKnights />} />
```

```
            <Route path='/host/turingtest' exact element={<HostTuringTest
/>} />
            <Route              path='/host/skibidyparty'              exact
element={<HostSkibidyParty />} />
            <Route path='/*' element={<Navigate replace to='/host/menu' />}
/>
          </Routes>
        )
      }
```

Цей код налаштовує маршрути наступним чином:

– за шляхом «/host/menu» доступний компонент HostChooseGame, що відповідає за сторінку вибору ігор;

– якщо шлях не визначений, користувача перенаправляє на «/host/menu»;

– за шляхом «/host/brainknights» доступний компонент HostBrainKnights, що відповідає за сторінку гри Brain Knights;

– за шляхом «/host/turingtest» доступний компонент HostTuringTest, що відповідає за сторінку гри Turing Test;

– за шляхом «/host/skibidyparty» доступний компонент HostSkibidyParty, що відповідає за сторінку гри Skibidy Party.

Для стилізації інтерфейсу в React було прийнято рішення використовувати SCSS. Це корисно, адже SCSS є препроцесором CSS, який дозволяє використовувати змінні, вкладені правила, міксіни та інші корисні функції, що покращують продуктивність розробки та підтримки стилів. Прикладом використання змінних для спрощення підтримки проєкту слугує наступний файл color.scss:

```
$black: #353535;
$white: #FFFFFF;
$brand1-color: #5279e4;
$brand2-color: #0066C3;
$brand3-color: #2F4858;
$neutral100: #EAEDEE;
$neutral200: #D5DADE;
$neutral300: #ACB6BC;
$neutral400: #445A69;
$neutral500: #263A46;
$neutral600: #0E161A;
$Senary: #DCDCDC;
$Info: #0064CF;
$Success: #1D5B40;
$Warning: #FF5C00;
$Info-Bkg: rgba(0, 101, 195, 0.1);
$Success-Bkg: rgba(47, 72, 88, 0.1);
```

```
$Warning-Bkg: rgba(228, 87, 82, 0.1);
$Error: #D51121;
$Error-Bkg: #d5112110;
$Light-Accent-Default: $brand1-color;
$Light-Accent-Hover: #D24E4B;
$Light-Accent-Disable: $neutral300;
$Light-1-Default: $black;
$Light-1-Hover: $neutral500;
$Light-1-Disable: $neutral300;
$Light-2-Default: $black;
$Light-2-Hover: $neutral500;
$Light-2-Disable: $neutral300;
$Light-2-Default: $neutral500;
$Light-2-Hover: rgba(53, 53, 53, 0.64);
$Blurred-Gray: rgba(92, 92, 92, 0.50);
$Text-Body: $neutral400;
$Text-Heading: $neutral600;
```

Його використання дозволяє централізовано керувати кольоровою палітрою проєкту. Якщо потрібно змінити колір будь-якого елемента, це можна зробити в одному місці, змінюючи значення відповідної змінної. Це значно полегшує підтримку та оновлення стилів у проєкті.

Управління локалізацією в цьому застосунку React реалізовано за допомогою бібліотеки i18next [12] наступним чином:

```
import i18n from 'i18next'
import { initReactI18next } from 'react-i18next'
import HttpApi from 'i18next-http-backend'
import LanguageDetector from 'i18next-browser-languagedetector'
i18n
      .use(HttpApi)
      .use(LanguageDetector)
      .use(initReactI18next)
      .init({
        fallbackLng: 'en',
        detection: {
          order: ['querystring', 'cookie', 'localStorage', 'navigator',
'htmlTag'],
          caches: ['cookie'],
        },
        interpolation: {
      })
export default i18n
```

У цьому коді відбувається інтеграція бібліотеки i18next з React застосунком. Крім того, тут налаштовується мова за замовчуванням, а також експортується ця конфігурація для використання в інших частинах додатку.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування має охоплювати всі розроблені функції клієнтської хост частини гри в рамках програмного ігрового застосунка для кваліфікаційної роботи, що включає логіку всіх трьох ігор, коректне відображення ігрового оточення та програмних інтерфейсів системи.

Для тестування було використано ручне тестування. За допомогою цього методу легко приділити більше уваги деталям та виявити критичні проблеми. Ручне тестування сприяє швидкому виявленню та виправленню помилок. Крім того ручне тестування надає можливість використовувати нестандартні кроки в тестуванні та творчий підхід.

Для ручного тестування ігрового програмного застосунку потрібно використовувати структурний підхід, для того, щоб охопити всі вимоги зазначені у специфікації. Тестування клієнтської частини програмного забезпечення складалося з наступних кроків:

а) планування: створено тест-план (див. додаток Г);

б) перевірка функціональних вимог:

1) для гри «Turing Test»:

– перевірено можливість змінювати команду користувачем;

– перевірено можливість змінити статус гравця на «готовий»;

– перевірено можливість почати гру;

– перевірено коректність відображення QR-коду для підключення в ігровому лобі гри;

– перевірено можливість почати гру;

– перевірено можливість поставити гру на паузу;

– перевірено коректність відображення питань та відповідей на хост частині клієнту;

– перевірено коректність відображення правильної та неправильної відповіді;

– перевірено, що відповіді відображаються у випадковому порядку;

– перевірено коректність відображення правильної та неправильної відповіді;

– перевірено загальну коректність відображення інтерфейсу в грі;

2) для гри «Skibidy Party»:

– перевірено коректність завантаження свого набору мемів до гри;

– перевірено можливість змінити статус гравця на «готовий»;

– перевірено можливість почати гру;

– перевірено коректність відображення QR-коду для підключення в ігровому лобі гри;

– перевірено коректність відображення статистики;

– перевірено коректність відображення обраних гравцями мемів;

– перевірено коректність відображення переможців;

– перевірено коректність відображення всіх інтерфейсів в грі;

– перевірено можливість поставити гру на паузу;

3) для гри «Brain Knights»:

– перевірено можливість змінювати команду користувачем;

– перевірено можливість змінити статус гравця на «готовий»;

– перевірено можливість почати гру;

– перевірено зміну ролі капітанів до іншого користувача;

– перевірено можливість поставити паузу за допомогою хост частині клієнту;

– перевірено коректність відображення екрану початку першого раунду;

– перевірено коректність відображення та зміни питань в бліц-раунді;

– перевірено коректність відображення правильних та неправильних питань в бліц-раунді;

– перевірено коректність відображення вибору теми для другого раунду гри;

– перевірено відображення правильних та неправильних питань в другому раунді гри;

3) клієнт загалом:

– перевірено, що виконується валідація отриманих даних;

– перевірено коректність програвання музики у всіх іграх та меню програмного застосунку;

– перевірено коректність програвання звукових ефектів у всіх іграх;

– перевірено функцію зміни екрану з хосту на контролер на клієнтській частині гри;

– перевірено функцію зміни мови у меню та у всіх іграх в меню програмного застосунку;

– перевірено, що досягнення розподіляються серед гравців;

в) перевірка нефункціональних вимог:

1) перевірено, що підтримується українська та англійська мови контенту, який є статичним для гри;

2) перевірено, що виконується логування помилок та дій гравців;

3) перевірено, що файли зображень зберігаються в локальному файловому сховищі на тому самому пристрої.

4) перевірено, що налаштування промпту для отримання відповіді від ШІ є коректними та відповідають очікуваним результатам..

В результаті проведеного тестування були створені тест-кейси для деяких функціональних вимог (див. додатку Д).

6 ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Впровадження програмного застосунку було здійснено шляхом розгортання проєкту на хмарній платформі Render, яка використовується для розгортання веб-застосунків, баз даних та серверних систем. Перед розгортанням застосунку було проведено підготовчий етап, який включає перевірку програмного застосунку шляхом розгортання на локальному сервері, зокрема клієнтскої частини. Також була проведена підготовка скриптів для управління життєвим циклом проєкту у файлі package.json. Було розроблено новий скрипт start (див. рис. 6.1), що виконує команду react-scripts start.

Рисунок 6.1 – Скрипти управління життєвим циклом проєкту у файлі package-json на клієнтскій частині застосунку (знімок екрана виконано самостійно)

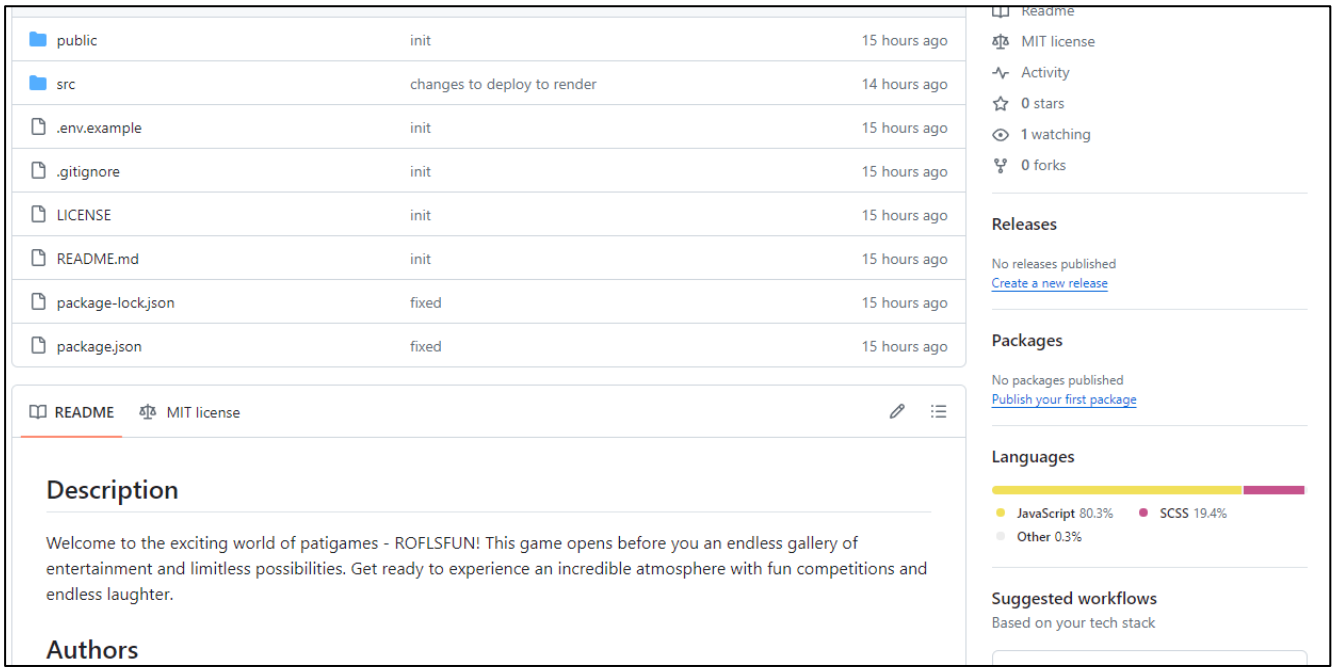


Рисунок 6.2 – GitHub-репозиторій з файлами клієнтскої частини проєкту (знімок екрана виконано самостійно)

Після цього до платформи Render було підключено GitHub-акаунт, створено безкоштовний веб-сервіс на базі Node.js та вказані команди для запуску й збірки проєкту (див. рис. 6.3).

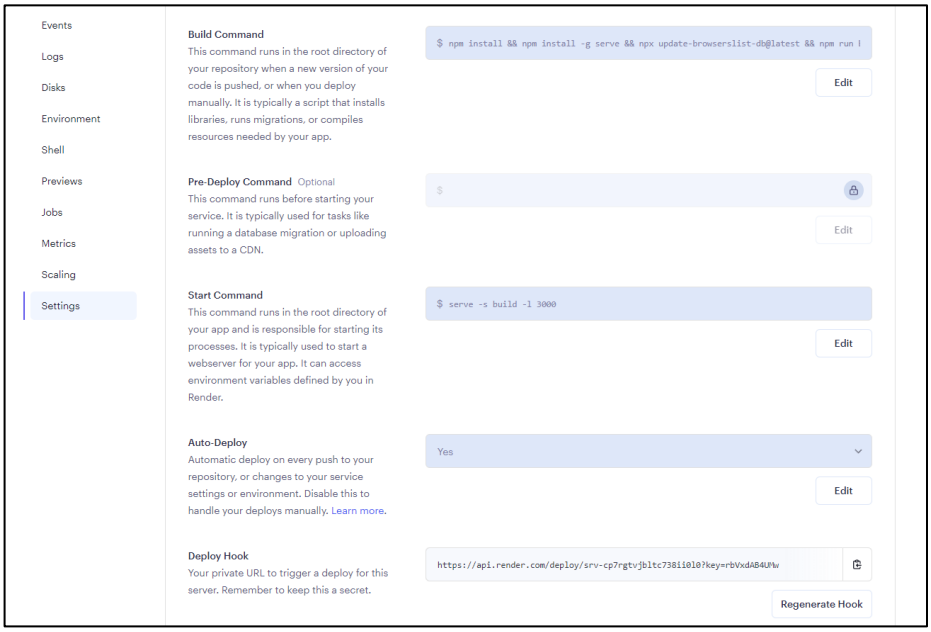


Рисунок 6.3 – Налаштування веб-сервісу на платформі Render (знімок екрана виконано самостійно)

Також було додано змінні середовища з файлу .env, який знаходився в локальному проєкті (див. рис. 6.4).

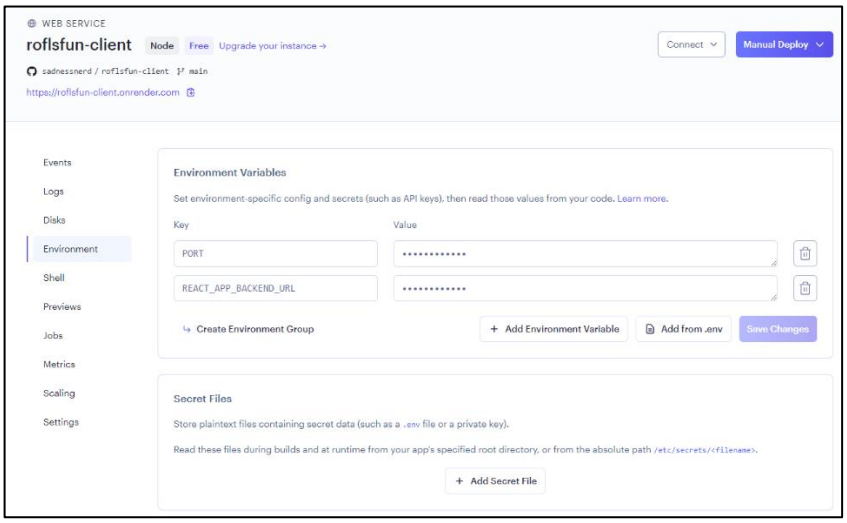


Рисунок 6.4 – Налаштування змінних середовища на платформі Render (знімок екрана виконано самостійно)

Після завершення розгортання проєкту на платформі Render, була перевірена коректна робота програмного застосунку методом перевірки роботи всіх сторінок в браузері.

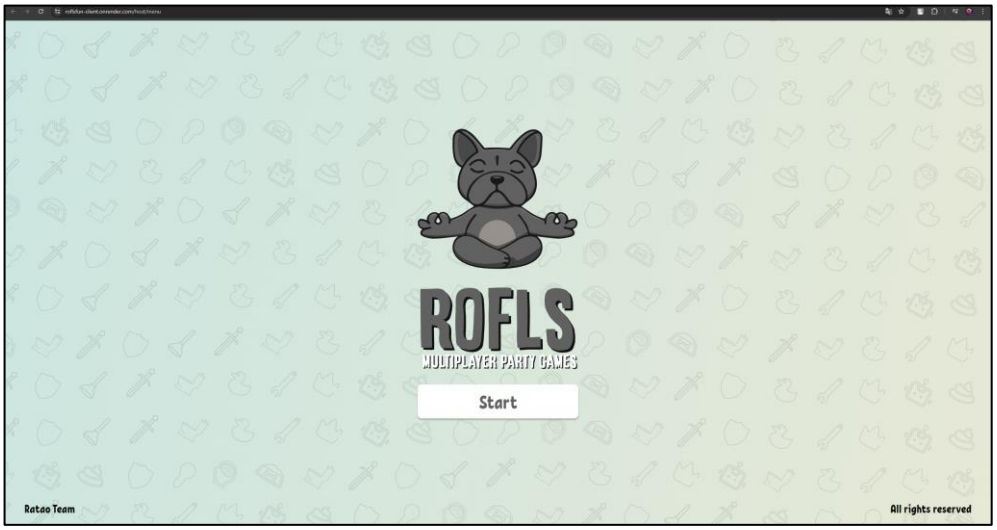


Рисунок 6.5 – Запит з клієнтської частини на розгорнуту back-end частину застосунку (знімок екрана виконано самостійно)

Таким чином, клієнтска частина ігрового програмного застосунку була впроваджена та протестована у мережі інтернет.

Також одним із способів провадження стала участь у конференції «Інформаційні інтелектуальні системи» на 28-му Міжнародному форумі «Радіоелектроніка та молодь у XXI столітті», де було представлено тези на тему «Використання смартфона з веб-застосунком як бездротового контролера в багатокористувацьких іграх на основі технології постійного підключення WEBSOCKET» (див. додаток Е). Цю роботу було відзначено під час роботи секції №3 «Програмна інженерія. Інформаційні технології в освіті» та нагороджено грамотою за змістовну доповідь (див. додаток Ж).

Ще одним із методів впровадження розробленого програмного забезпечення стала участь у виставці технічної творчості молоді під час 28-го Міжнародного форуму «Радіоелектроніка та молодь у XXI столітті» (див. додаток И). На цій виставці команда проєкту ROFLSUN презентувала ігровий застосунок у жанрі multiplayer party games і здобула друге місце (див. додаток К).

ВИСНОВКИ

Під час виконання передатестаційної роботи бакалавру було створено ігровий програмний застосунок у жанрі multiplayer party games: хост частина, level-design та UI/UX

Хост екран клієнтської частини та геймплей було розроблено на клієнтській частині застосунку з використанням мови програмування JavaScript та бібліотеки React. Для управління станом даних на клієнтській стороні застосунку використовувалась бібліотека Redux, а для забезпечення обміну даними між клієнтською та серверною частинами в реальному часі – бібліотека Socket.IO на клієнтській частині проєкту. Для проєктування ігрового оточення був використаний сервіс для розробки інтерфейсів Figma.

Перед початком розробки системи було проведено проєктування архітектури та UML-моделювання серверної частини програмного забезпечення. Це дозволило чітко визначити структуру та взаємозв'язки між різними компонентами системи і, в результаті, сприяло успішному втіленню проєкту.

Основні результати роботи можуть бути узагальнені наступним чином:

– створення хост екрану клієнтскої частини гри для відтворення геймплею на ньому. Швидкий відгук на команди від контролера за допомогою мінімізації навантаження на серверну частину, методом перенесення основних графічних на хост частину та геймплейних на серверну частину проєкту;

– створення ігрового оточення: під час розробки було створено ігрове оточення для основного екрану гри та мобільного контролера, яке було проєктоване з урахуванням потреб користувачів і забезпечувало їм зручне та інтуїтивно зрозуміле використання. Для проєктування ігрового оточення використовувалась платформа для проєктування інтерфейсів Figma.

Отже, результати роботи свідчать про успішну реалізацію поставлених завдань та теми. Можна зробити висновок, що мета пре роботи була досягнута, а ігровий програмний застосунок у жанрі multiplayer party games готовий до

використання. Гра призначена для широкого спектру користувачів, це досягнуто, зокрема, завдяки правильно побудованому ігровому контролеру та зручно спроєктованого ігрового оточення. Гра запускається на будь-яких пристроях де є браузер с підтримкою JavaScript, що ще більше розширяє кількість можливих користувачів, через простоту використання додатку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Rauschmayer A. JavaScript for impatient programmers. Independently published, 2019. 526 с.

2. Elrom E. React and Libraries. Berkeley, CA : Apress, 2021. URL: https://doi.org/10.1007/978-1-4842-6696-0 (дата звернення: 25.05.2024).

3. Index | node.js v20.13.1 documentation. *Node.js – Run JavaScript Everywhere*. URL: https://nodejs.org/docs/latest-v20.x/api/ (дата звернення: 25.05.2024).

4. Getting started with redux | redux. *Redux - A JS library for predictable and maintainable global state management | Redux*. URL: https://redux.js.org/introduction/getting-started (дата звернення: 25.05.2024).

5. Learning UML 2.0: A Pragmatic Introduction to UML. O'Reilly, 2006. 228 с.

6. Salcescu C. Functional Architecture with React and Redux. Independently Published, 2020. 148 с.

7. Двугрошев А. О. Використання смартфона з веб-застосунком як бездротового контролера в багатокористувацьких іграх на основі технології постійного підключення WebSocket / А. О. Двугрошев // Радіоелектроніка та молодь у XXI столітті: тези доповідей 28-го Міжнародного молодіжного форуму, 16–18 квітня 2024 р. – Харків : ХНУРЕ, 2024. – Т. 6. – С. 369–371.

8. Pronina D., Kyrychenko I. Comparison of Redux and React Hooks Methods in Terms of Performance / D. Pronina, I. Kyrychenko // Proceedings of the 6th International Conference on Computational Linguistics and Intelligent Systems (COLINS 2022). Volume I: Main Conference, 12–13 May 2022. – Gliwice : CEUR Workshop Proceedings, 2022. – № 3171. – Pp. 791–800.

9. Ареф’єв О. О. Інтерфейс користувача. Особливості розробки інтерфейсу користувача для ігрових застосунків / О. О. Ареф’єв // Радіоелектроніка та молодь у XXI столітті: тези доповідей 25-го Міжнародного молодіжного форуму, 20–22 квітня 2021 р. – Харків : ХНУРЕ, 2021. – Т. 6. – С. 361–362.

10. Introduction | socket.io. *Socket.IO*. URL: https://socket.io/docs/v4/ (дата звернення: 25.05.2024).

11. Xia J. UI/UX Design. Artpower International Publish Company, Limited, 2016. 256 с.

12. Introduction | i18next documentation. *Introduction | i18next documentation*. URL: https://www.i18next.com/ (дата звернення: 25.05.2024).

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

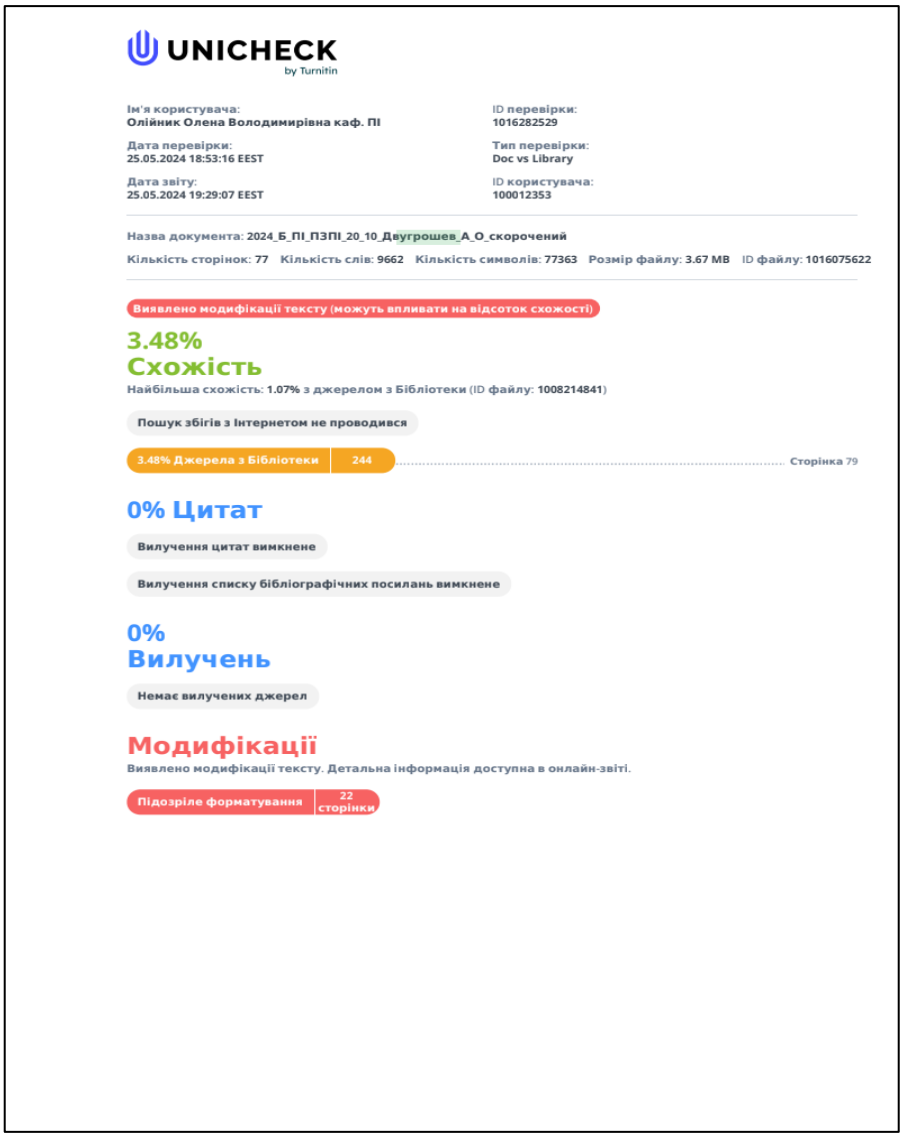


Рисунок А.1 Результати перевірки на унікальність тексту в базі ХНУРЕ (знімок виконано самостійно)

ДОДАТОК Б

Слайди презентації

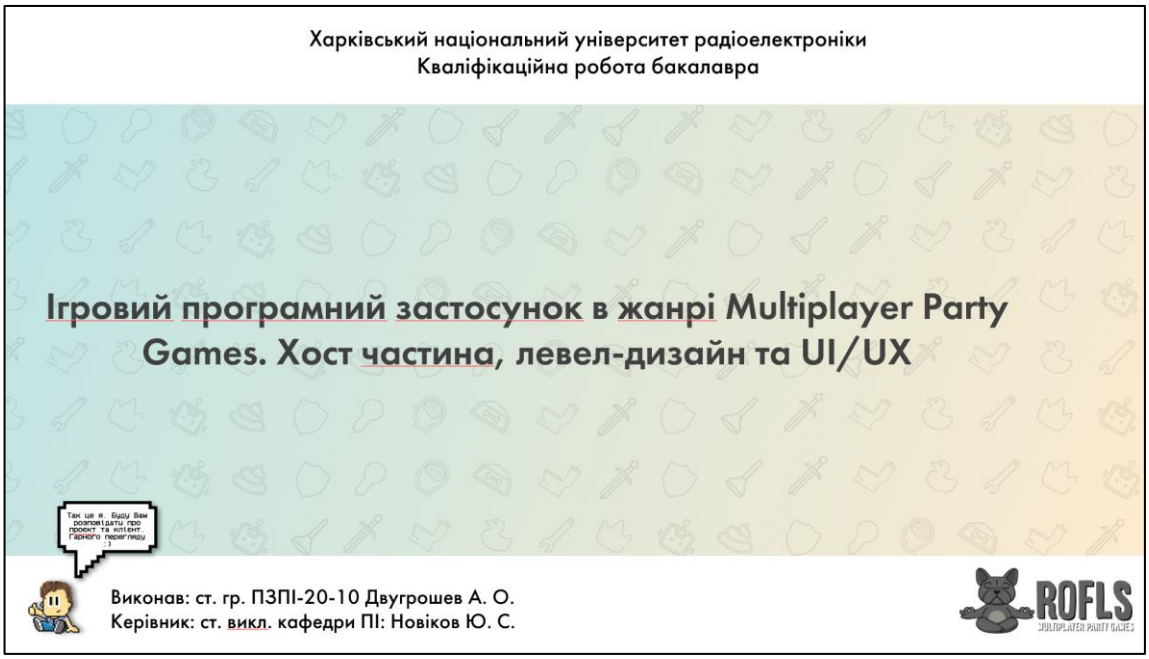


Рисунок Б.1 – Перший слайд презентації (знімок екрана виконано самостійно)

Рисунок Б.1 – Другий слайд презентації (знімок екрана виконано самостійно)

Рисунок Б.1 – Третій слайд презентації (знімок екрана виконано самостійно)

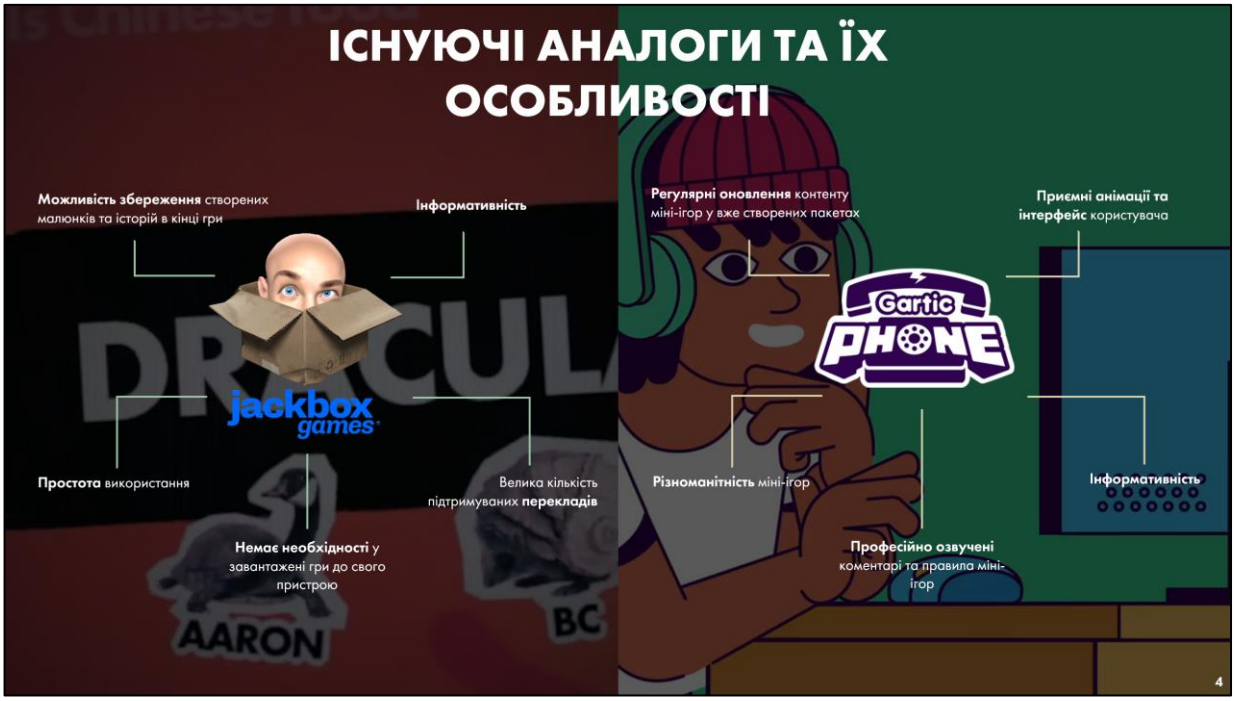


Рисунок Б.1 – Четвертий слайд презентації (знімок екрана виконано самостійно)

Рисунок Б.5 – П’ятий слайд презентації (знімок екрана виконано самостійно)

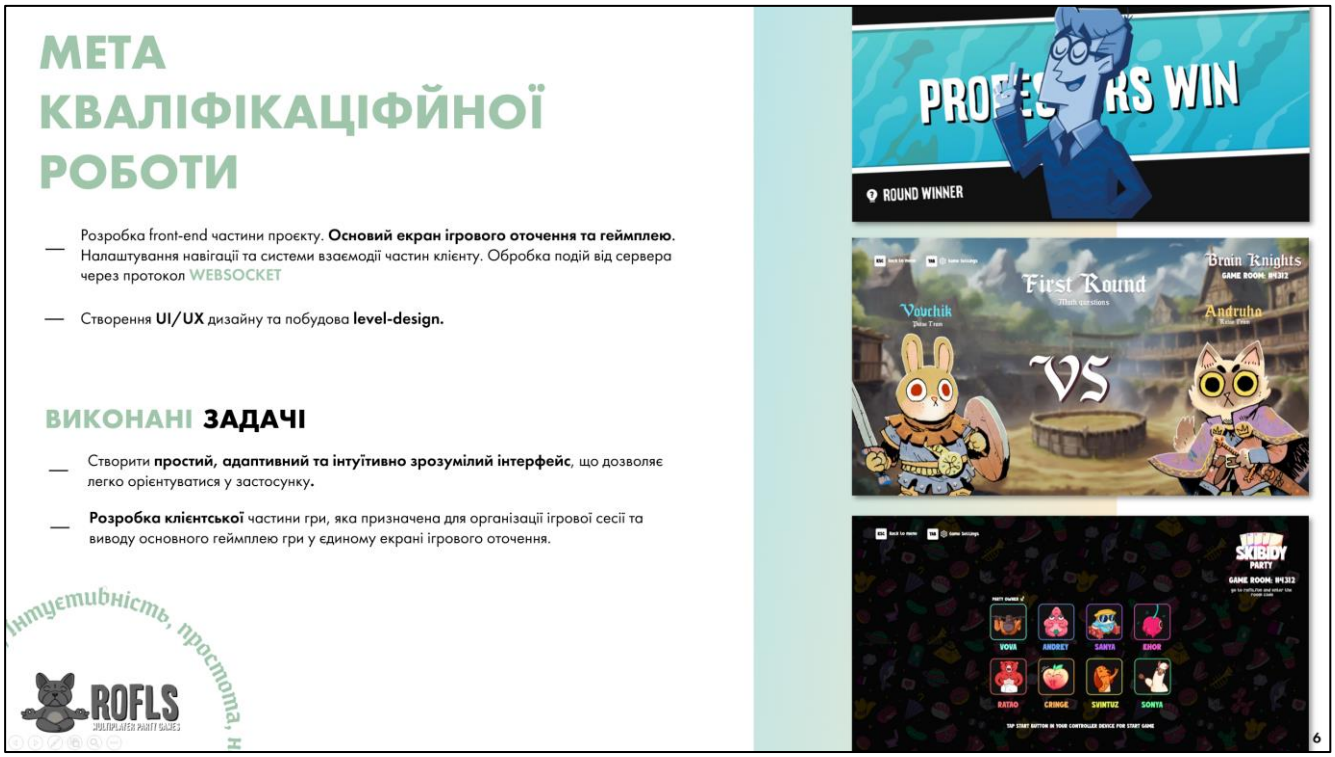


Рисунок Б.6 – Шостий слайд презентації (знімок екрана виконано самостійно)

Рисунок Б.7 – Сьомий слайд презентації (знімок екрана виконано самостійно)

Рисунок Б.8 – Восьмий слайд презентації (знімок екрана виконано самостійно)

Рисунок Б.9 – Дев’ятий слайд презентації (знімок екрана виконано самостійно)

Рисунок Б.10 – Десятий слайд презентації (знімок екрана виконано самостійно)

Рисунок Б.11 – Одинадцятий слайд презентації (знімок екрана виконано самостійно)

Рисунок Б.12 – Дванадцятий слайд презентації (знімок екрана виконано самостійно)

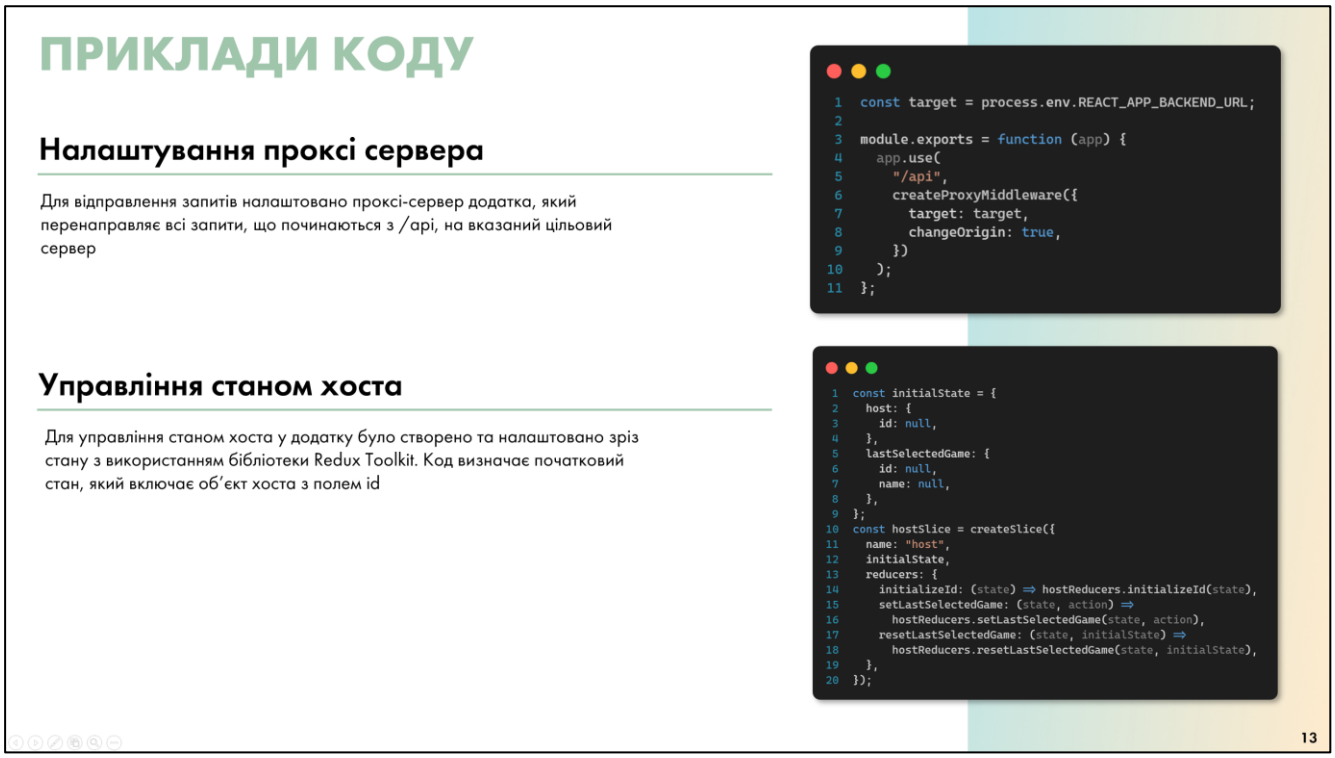


Рисунок Б.13 – Тринадцятий слайд презентації (знімок екрана виконано самостійно)

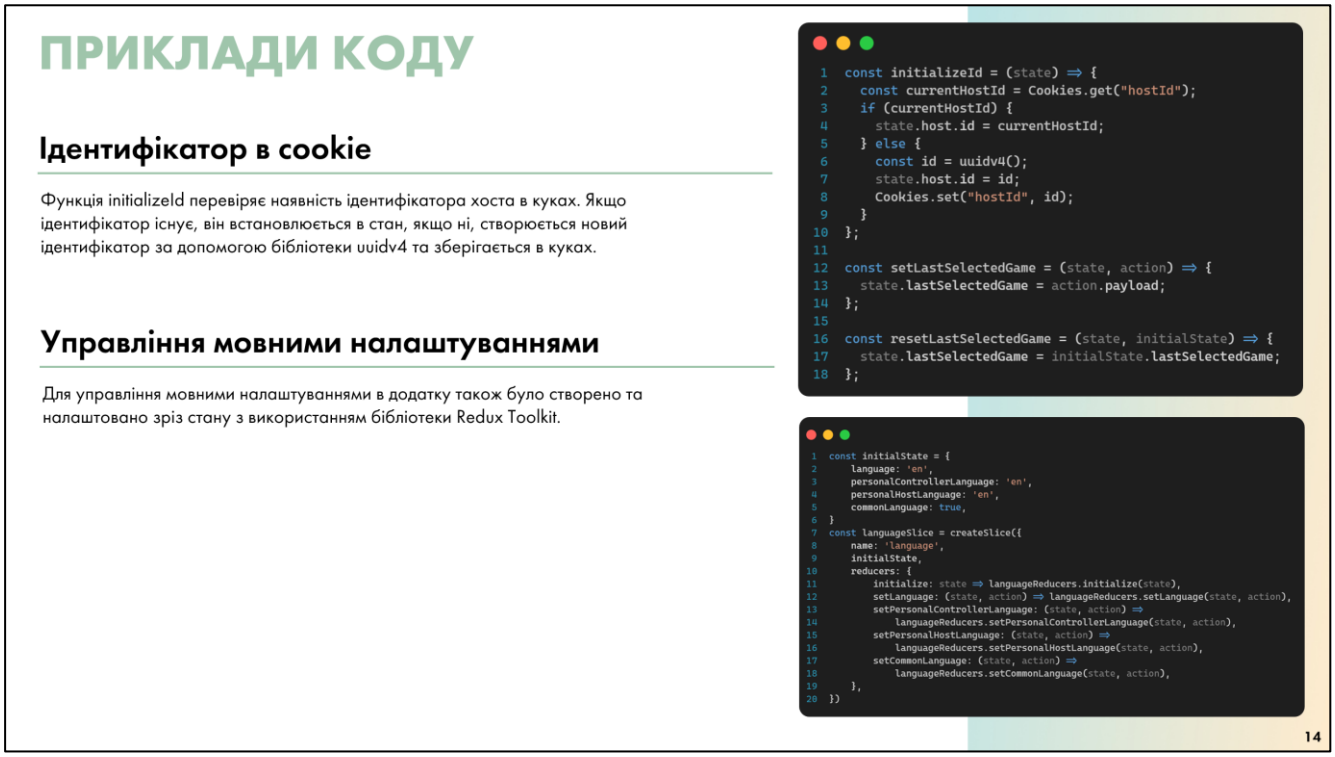


Рисунок Б.14 – Чотирнадцятий слайд презентації (знімок екрана виконано самостійно)

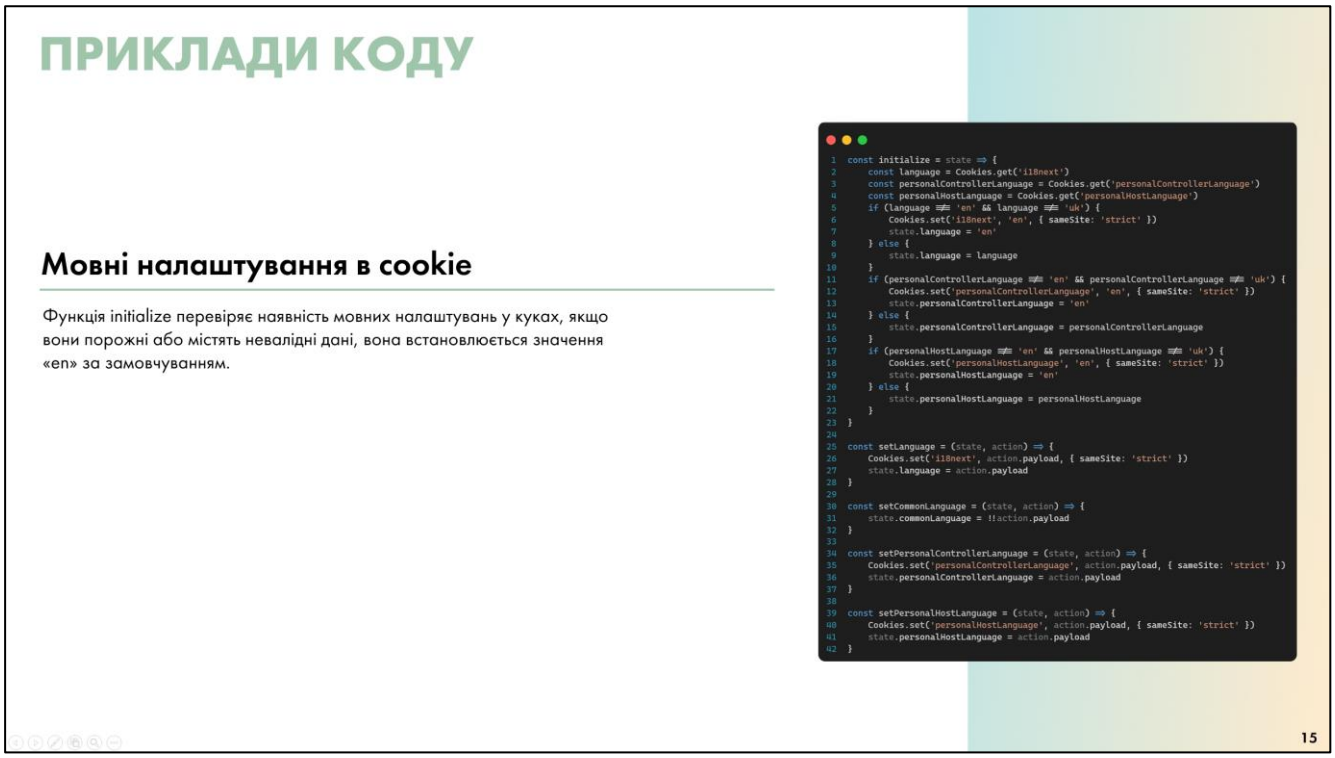


Рисунок Б.15 – П’ятнадцятий слайд презентації (знімок екрана виконано самостійно)

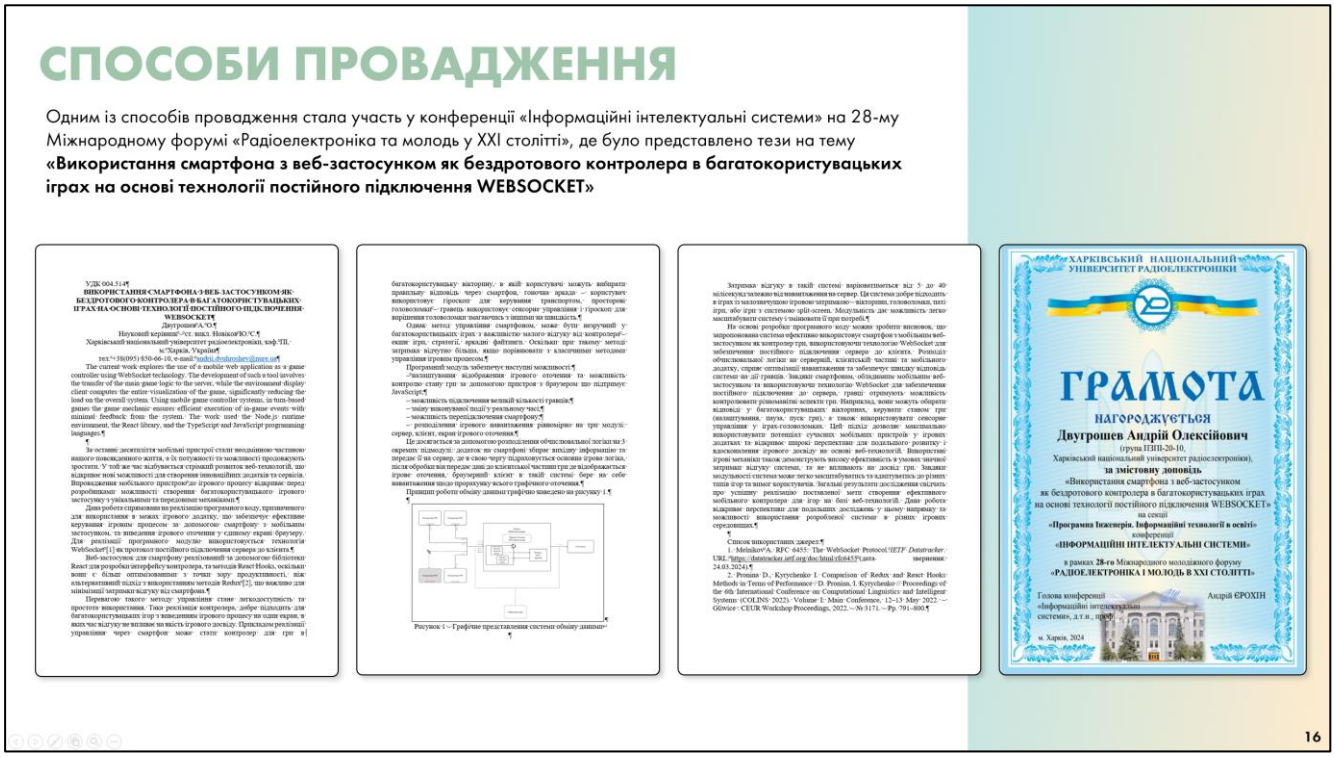


Рисунок Б.16 – Шістнадцятий слайд презентації (знімок екрана виконано самостійно)

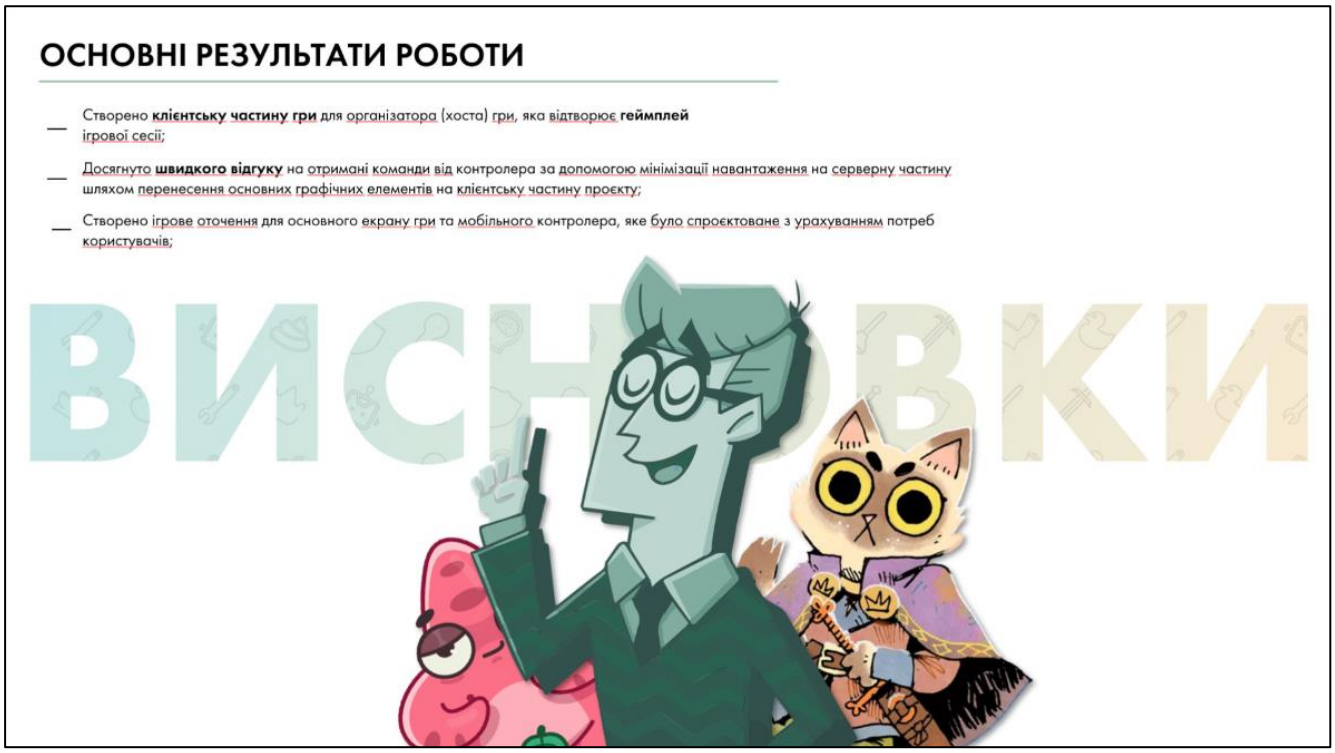


Рисунок Б.17 – Сімнадцятий слайд презентації (знімок екрана виконано самостійно)

ДОДАТОК В

Специфікація ігрового програмного застосунку у жанрі multiplayer party games

ROFLSFUN

Специфікація вимог до програмного забезпечення

1.0

15.05.2024

Команда проєкту ROFLSFUN

Історія версій

Дата	Опис	Автор	Коментарі
15.05.2024	Версія 1.0	Команда проєкту ROFLSFUN	Перша редакція

Затвердження документів

Наступна специфікація вимог до програмного забезпечення була прийнята та схвалена наступними особами:

Підпис	Друковане ім’я	Заголовок	Дата
	О. Ю. Калініченко	Back-end частина, інтеграція AI до системи	15.05.2024
	В. О. Бухало	Back-end частина, налаштування мережевої гри	15.05.2024
	А. О. Двугрошев	Хост частина, левел-дизайн та UI/UX	15.05.2024
	Є. В. Мірошніков	Контролер, баланс ігрового процесу	15.05.2024

ЗМІСТ

1 ВСТУП

1.1 Огляд продукту

Програмний продукт для якого розробляється специфікація – це ігровий програмний застосунок у жанрі multiplayer party games, геймплей якого відбуватиметься в онлайн-середовищі. Гра буде організовуватися під час особистих або віртуальних зустрічей з друзями з метою весело провести час. Ігровий програмний застосунок включатиме набір простих ігор наступних жанрів: гумористична карткова гра, соціальна дедуктивна гра та змагальна вікторина. Ці ігри спрямовані на соціальну взаємодію та розваги.

Гра жанру гумористична карткова гра буде складатися з кількох раундів, кількість яких буде залежати від кількості учасників, що може варіюватися від чотирьох до восьми. На кожному раунді випадковим чином обиратиметься суддя з тих учасників, які раніше не виконували цю роль. Суддя оцінюватиме меми, які вибрали звичайні гравці, відповідно до випадково обраної ситуації. Звичайні гравці отримуватимуть по шість унікальних карток з мемами, які не повторюватимуться серед карток інших гравців у поточному або попередніх раундах. Після цього суддя отримає список мемів, обраних звичайними гравцями, і буде ранжувати їх від першого до третього місця. Гравець, який займе третє місце, отримає 500 балів, друге місце призведе до отримання 1000 балів, а переможець, здобувши перше місце, отримає 1500 балів. Переможцем стане той, хто набере найбільшу кількість балів протягом усіх раундів гри.

Соціальна дедуктивна гра буде складатися з трьох раундів, і в ній зможуть брати участь від чотирьох до восьми гравців. У грі гравці розподілятимуться на команди професорів і студентів в залежності від їх кількості за такими пропорціями: при чотирьох гравцях – один студент і три професори; при п’яти гравцях – два студенти і три професори; при шести гравцях – два студенти та чотири професори; при семи гравцях – три студенти і чотири професори; при восьми гравцях – три студенти і п’ять професорів. Перед початком гри присутні

в кімнаті гравці зможуть змінити свою команду, якщо кількість вільних місць це дозволяє. Гра розпочнеться з команди студентів, кожен з них задаватиме питання, на яке повинні дати відповідь професори та штучний інтелект. Після відповідей професорів, штучний інтелект отримає дані і намагатиметься сформувати свою відповідь так, щоб вона була схожа на відповідь професорів. Після цього настане етап голосування, під час якого студенти обиратимуть відповідь, яка, на їхню думку, надана штучним інтелектом. Якщо студенти вгадають, яку відповідь надав штучний інтелект, їхня загальна сума балів збільшиться на два. В іншому випадку один бал додасться до загальної суми балів професорів. Команда, яка набере більше балів, переможе. Якщо набрано рівну кількість балів, то гра завершиться внічию. Команда студентів має на меті виявлення штучного інтелекту серед професорів. Команда професорів намагається заплутати студентів, виступаючи як штучний інтелект.

Гра жанру змагальна вікторина складатиметься з двох етапів та буде розрахована для участі від чотирьох до восьми гравців. Перший етап передбачатиме проведення низки раундів. Кількість раундів визначатиметься як подвоєна максимальна кількість гравців однієї з двох команд, щоб кожен гравець мав можливість взяти участь принаймні двічі. Тема обиратиметься випадковим чином кожен раунд, після чого капітан команди вибиратиме гравця, який відповість на питання з цієї теми. Кожне питання матиме чотири варіанти відповідей, один з яких буде правильним. Система відстежуватиме кількість правильних відповідей, і в кінці кожного раунду учасник, який набрав найбільше правильних відповідей, приєднуватиме один бал до загального командного рахунку. У випадку рівної кількості правильних відповідей, обидва учасники отримуватимуть по одному балу. Після завершення першого етапу визначатиметься найкращий гравець гри, який набрав найбільше правильних відповідей. Другий етап – це командна вікторина, у якій братимуть участь всі члени команди. Цей етап складатиметься з трьох раундів, під час кожного з яких капітани обиратимуть чотири теми. Запитання ставатимуть складнішими, а командам надаватиметься достатньо часу на обговорення питання з однією правильною відповіддю. Кожна правильна

відповідь принесе команді один бал. Після завершення другого етапу визначатиметься переможна команда, яка здобула найбільше балів. Якщо бали однакові, оголошуватиметься нічия.

У межах цього ігрового застосунку гравці матимуть чіткий розподіл ролей, де одні транслюватимуть ігровий процес, а інші керуватимуть ним. Гравці будуть керувати ігровим процесом за допомогою своїх смартфонів, планшетів або інших цифрових пристроїв з доступом до Інтернету та веб-браузера. Трансляція ігрового процесу відбудеться за допомогою тих самих пристроїв, що і для керування, проте варто враховувати, що бажано, щоб він транслювався на великому екрані, щоб всім гравцям було зручно спостерігати за ним. Також, гравці, які транслюватимуть ігровий процес, матимуть змогу виконувати налаштування ігрового процесу, включаючи налаштування звукового супроводу та локалізації, що включатиме англійську та українську мови.

Ігровий програмний застосунок складатиметься з двох складових: серверної та клієнтської частин. Серверна частина системи буде забезпечувати управління грою, синхронізацію дій між усіма гравцями та обробку та збереження даних гри протягом ігрової сесії. Клієнтська частина буде забезпечувати зручний та візуально привабливий інтерфейс, що відображатиме дані гри та дозволятиме користувачам керувати ігровим процесом.

Щодо інтеграції штучного інтелекту для соціально дедуктивної гри, вона відбудеться за допомогою API компанії OpenAI, що надасть доступ до використання моделі GPT Text-Davinci-003.

1.2 Мета

Метою цього SRS документу є надання докладного опису вимог до програмного продукту. Цей документ призначений для всіх, хто буде приймати участь у програмній реалізації проєкту та його тестуванні, зокрема для розробників

клієнтської та серверної частин, а також для тестувальників, з метою чіткого усвідомлення функціональних та нефункціональних вимог до системи.

1.3 Межі

Програмний продукт, для якого розробляється специфікація, можна ідентифікувати як ROFLSFUN. Його ігри можна ідентифікувати наступним чином: гумористична карткова гра Skibidy Party, соціальна дедуктивна гра Turing Test та змагальна вікторина Brain Knights.

Програмний продукт ROFLSFUN буде підтримувати:

– українську локалізацію;

– англійську локалізацію;

– інтерфейс для гравців, які транслюють ігровий процес;

– інтерфейс для гравців, які керують ігровим процесом;

– функціонал гри Skibidy Party;

– функціонал гри Turing Test;

– функціонал гри Brain Knights;

– загальний функціонал застосунку, такий як ідентифікація гравця, пошук кімнати, ініціалізація ігрової сесії, зміна налаштувань гри.

Ігровий програмний застосунок ROFLSFUN повинен бути розроблений для забезпечення інтерактивного, багатокористувацького ігрового досвіду у жанрі multiplayer party games, що спрямований на створення веселих та соціально інтерактивних ситуацій під час особистих або віртуальних зустрічей з друзями.

Переваги цього програмного продукту:

– ROFLSFUN сприятиме активній соціальній взаємодії між гравцями, залучаючи їх до спільних ігрових активностей, що буде зміцнювати дружні зв’язки та сприяти веселощам;

– програмне забезпечення дозволятиме гравцям брати участь у грі з різних цифрових пристроїв (смартфони, планшети, ПК) з доступом до Інтернету та наявністю веб-браузера, що буде робити його доступним та зручним для користувачів;

– гравці будуть мати можливість обирати як англійську, так і українську локалізації;

– у гру Turing Test буде інтегровано ШІ, що дозволить генерувати реалістичні відповіді на основі переданої підказки; для інтеграції буде використано API компанії OpenAI, що надасть доступ до використання моделі GPT Text-Davinci-003; відповіді, згенеровані за допомогою цієї моделі, будуть реалістичними, що сприятиме цікавості до гри.

Завдання цього програмного продукту:

– надати інтуїтивно зрозумілий інтерфейс, який дозволить легко керувати ігровим процесом, а також переглядати результати цього керування;

– забезпечити синхронізацію дій між гравцями та обробку даних гри в реальному часі;

– забезпечити підтримку української та англійської локалізацій;

– забезпечити правильну взаємодію зі API компанії OpenAI.

Цілі цього програмного продукту:

– створити ігрове середовище, яке сприяє активній соціальній взаємодії, де гравці можуть насолоджуватися веселими та цікавими ігровими активностями разом з друзями або знайомими;

– надати гравцям можливість вибирати різні ігри з різних жанрів;

– забезпечити простоту та зручність використання, забезпечуючи гравцям легкий доступ до ігрових функцій та інтерфейсу.

Сфера застосування програмного продукту ROFLSFUN передбачає використання його під час соціальних заходів, як особистих, так і віртуальних, для покращення комунікації та взаємодії між учасниками через захопливий ігровий процес.

1.4 Посилання

В SRS документі відсутній перелік документів, на які є посилання в інших його частинах або в окремому документі, визначеному специфікацією. Також, SRS документ не містить жодних інших посилань.

1.5 Означення та абревіатури

API – інтерфейс програмування застосунків
GPT – Generative Pre-trained Transformer
SRS – специфікація вимог до програмного забезпечення
ШІ – штучний інтелект
DOM – об'єктна модель документа
БД – база даних
HTML – мова розмітки гіпертексту
CSS – каскадні таблиці стилів
QR – швидка відповідь
HTTP – протокол передачі гіпертексту
HTTPS – захищений протокол перед

2 ЗАГАЛЬНИЙ ОПИС

2.1 Перспективи продукту

ROFLSFUN планується як ігровий програмний застосунок у жанрі multiplayer party games, що забезпечуватиме інтерактивний ігровий досвід в онлайн-середовищі. Аналогами такого застосунку є Jackbox Party Pack і Gartic Phone, які вже заслужили попит на ринку ігор у своєму жанрі.

Jackbox Party Pack включає набір міні-ігор, орієнтованих на соціальну взаємодію і розваги. Ключовими особливостями цього продукту є:

– кожен випуск Jackbox Party Pack містить кілька різних ігор, що дозволяє гравцям вибирати за інтересами;

– ігри доступні на багатьох платформах, включаючи ПК, консолі та мобільні пристрої;

– гравці можуть використовувати свої смартфони або планшети як контролери для гри.

Gartic Phone – це безкоштовна онлайн-гра, заснована на класичній грі «зіпсований телефон», де гравці малюють і вгадують малюнки один одного. Основні особливості Gartic Phone включають:

– гра дуже проста у використанні, з низьким порогом входження;

– основний геймплей орієнтований на малювання та творчі завдання;

– гра доступна безкоштовно в веб-браузері, що робить її легко доступною для всіх користувачів.

В свою чергу, ROFLSFUN пропонуватиме наступні особливості:

– різноманітність ігор різних жанрів: карткова гра, соціальна дедуктивна гра та змагальна вікторина. Це дозволить конкурувати з Jackbox Party Pack, надаючи користувачам ширший вибір розваг;

– підтримку багатомовності: англійська та українська мови;

– безкоштовну доступність у веб-браузері, що робитиме ROFLSFUN легко доступним для всіх користувачів.

Таким чином, ROFLSFUN матиме значний потенціал для розвитку, оскільки він поєднуватиме в собі особливості ігор, які вже заслужили попит на ринку, такі як різноманітність ігор різних жанрів, підтримку декількох мов та доступність. Це робитиме його привабливим вибором для користувачів, які шукають ігровий додаток для отримання нового ігрового досвіду у жанрі multiplayer party games.

2.2 Функції продукту

Стислий опис загальних функцій, виконання яких забезпечуватиме програмне забезпечення:

– зміна мови гри та гучності звукових ефектів;
– ініціалізація ігрової сесії;
– пауза та відновлення ігрової сесії;
– підключення до ігрової сесії;
– вихід з ігрової сесії;
– завершення ігрової сесії;

Стислий опис функцій в грі Skibidy Party, виконання яких забезпечуватиме програмне забезпечення:

– проведення раундів, генерація судді, випадкової ситуації та мемів;
– вибір та оцінка мемів;
– розподіл балів гравцям за перше, друге і третє місця;
– визначення переможця раунду та гри.

Стислий опис функцій в грі Turing Test, виконання яких забезпечуватиме програмне забезпечення:

– розподіл гравців на команди студентів і професорів;
– проведення раундів з питаннями та відповідями;
– генерація відповідей з допомогою ШІ;

– голосування студентів для виявлення відповіді штучного інтелекту;

– нарахування балів командам та визначення переможця.

Стислий опис функцій в грі Brain Knights, виконання яких забезпечуватиме програмне забезпечення:

– розподіл гравців на команди, надання ролі капітана;

– проведення першого етапу з індивідуальними питаннями;

– обробку наданих відповідей;

– нарахування балів за правильні відповіді;

– визначення найкращого гравця;

– проведення другого етапу з командними питаннями;

– нарахування балів командам та визначення переможця.

Таким чином, програмне забезпечення забезпечуватиме роботу програмного ігрового застосунку в цілому, а також ігор Turing Test, Skibidy Party та Brain Knights у ньому.

2.3 Характеристики користувачів

Демографічні характеристики: передбачається, що основна цільова аудиторія буде складатися з молоді та дорослих у віковому діапазоні від 18 до 40 років, користувачі можуть бути з будь-якої частини світу, але основний акцент робиться на англомовних та україномовних регіонах. Щодо технічних навичок та досвіду, передбачається, що користувачі матимуть базові навички роботи з комп’ютерами та мобільними пристроями, включаючи використання веб-браузерів та додатків, а також різний рівень досвіду у відеоіграх, від початківців до досвідчених гравців. Щодо інтересів та мотивації, передбачається, що користувачі будуть зацікавлені в іграх, які сприяють соціальній взаємодії та командній роботі, зокрема в умовах особистих або віртуальних зустрічей з друзями, а також мотивом для їх гри буде веселе проведення часу та створення приємної атмосфери під час зустрічей.

2.4 Загальні обмеження

Для реалізації фронтенду використовується бібліотека React та мова програмування JavaScript. Це обумовлено тим, що React надає великий набір ресурсів для розробки веб-додатків. Веб-додатки, створені з використанням цієї бібліотеки, відзначаються високою продуктивністю, оскільки React взаємодіє зі своїм легковажним еквівалентом – віртуальним DOM, замість повільних та незручних взаємодій безпосередньо з реальним DOM. Реальний DOM оновлюється лише після взаємодії з віртуальним DOM. Також можлива повторна використання компонентів, що скорочує час розробки. Використання JavaScript як мови програмування для фронтенду є універсальним рішенням для створення динамічних і інтерактивних веб-додатків, забезпечуючи швидку розробку та легку інтеграцію з React.

Для реалізації бекенду використовується середовище Node.js та фреймворк NestJS на мові програмування TypeScript. Це обумовлено тим, що серверна частина, написана за допомогою Node.js, має високу продуктивність. Наявність асинхронних бібліотек є дуже корисною, оскільки сервери Node.js не чекають відповіді від API, а переходять до наступного запиту. Використання фреймворка NestJS забезпечує модульну архітектуру, яка полегшує розробку, тестування та підтримку коду. TypeScript додає статичну типізацію, що допомагає уникати багатьох помилок на етапі розробки.

Для забезпечення безперервного зв'язку на бекенді використовуються WebSockets, а на фронтенді – Socket.IO. Це обумовлено тим, що ці технології дозволяють створювати додатки реального часу, забезпечуючи постійне з’єднання між клієнтом і сервером. Це є ключовим для роботи з ігровими сесіями, де потрібна швидка та постійна передача даних.

Система не використовує БД для зберігання інформації. Дані ігрових сесій зберігаються безпосередньо в пам'яті сервера, що забезпечує швидкий доступ і обробку тимчасових даних ігрової сесії.

2.5 Припущення й залежності

Для коректної роботи веб-додатку на пристрої потрібна підтримка HTML5 та CSS3, а також стабільне Інтернет-підключення з достатньою швидкістю завантаження і високою пропускною здатністю для плавної роботи програми. Щоб використовувати додаток як хосту, потрібно запустити його на комп’ютері або консолі. Для використання додатку як контролеру, необхідно мати смартфон або будь-який інший пристрій.

3 КОНКРЕТНІ ВИМОГИ

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

При вході користувача на сайт через хост, він переходить на головну сторінку (рис. 3.1), де може натиснути кнопку «Start», щоб перейти до меню вибору міні-ігор.

Рисунок 3.1 – Інтерфейс стартового екрану хоста (знімок екрана виконано самостійно)

При вході користувача на сайт через контролер, він переходить на головну сторінку (рис. 3.2), де може ввести особистий нікнейм та код кімнати. Якщо код кімнати дійсний, з’явиться кнопка з назвою міні-гри (Brain Knights, Turing Test або Skibidy Party), яка дозволить під’єднатися до неї.

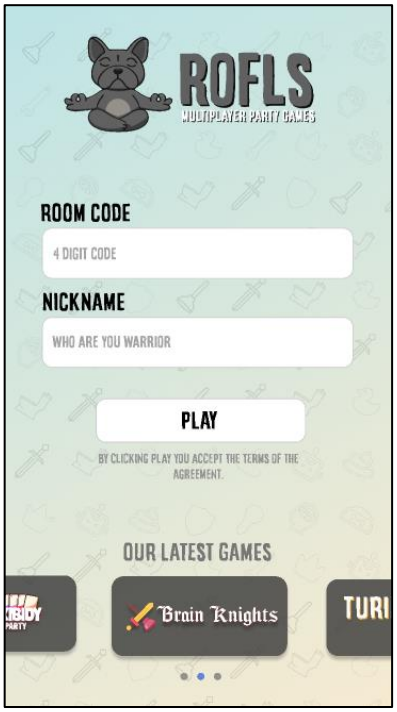


Рисунок 3.2 – Інтерфейс стартового екрану контролера (знімок екрана виконано самостійно)

Після переходу користувача до меню вибору міні-ігор, він потрапляє на сторінку (рис. 3.3-3.5), де може обрати одну з запропонованих міні-ігор, користуючись слайдером. Після вибору гри він може натиснути кнопку «Start Game», щоб створити приватну ігрову кімнату; кнопку «Game settings», щоб налаштувати ігровий процес (рис. 3.6); а також кнопку «Back to main screen», щоб повернутися до головного меню.

Рисунок 3.3 – Вибір гри Brain Knights (знімок екрана виконано самостійно)

Перший слайд відображає гру Brain Knights разом з коротким описом самої гри.

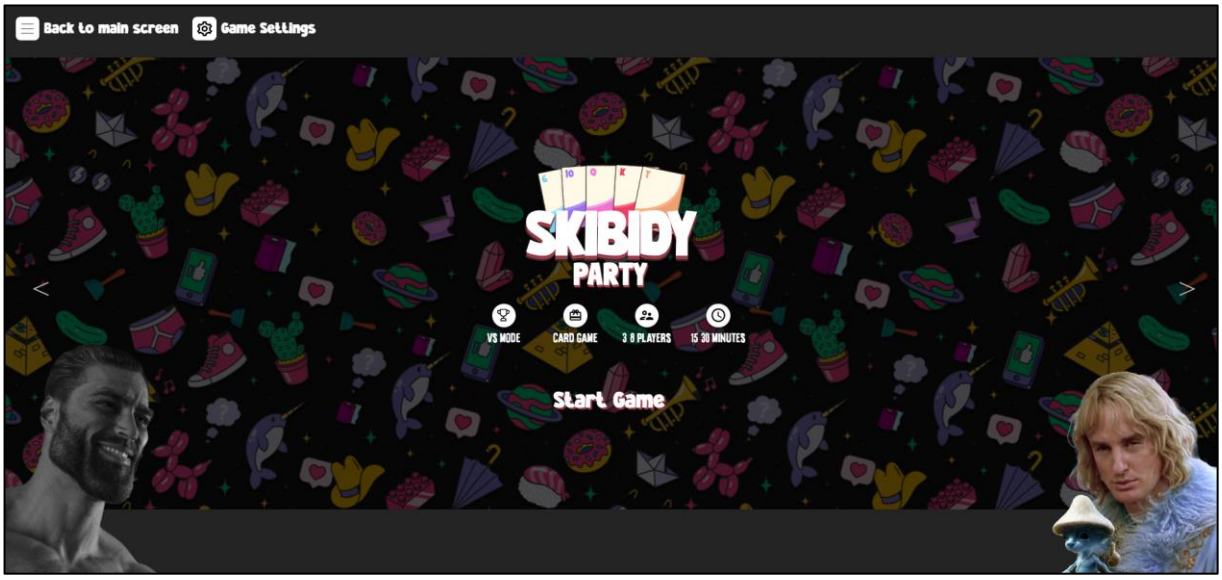


Рисунок 3.4 – Вибір гри Skibidy Party (знімок екрана виконано самостійно)

Другий слайд відображає гру Skibidy Party разом з коротким описом самої гри.

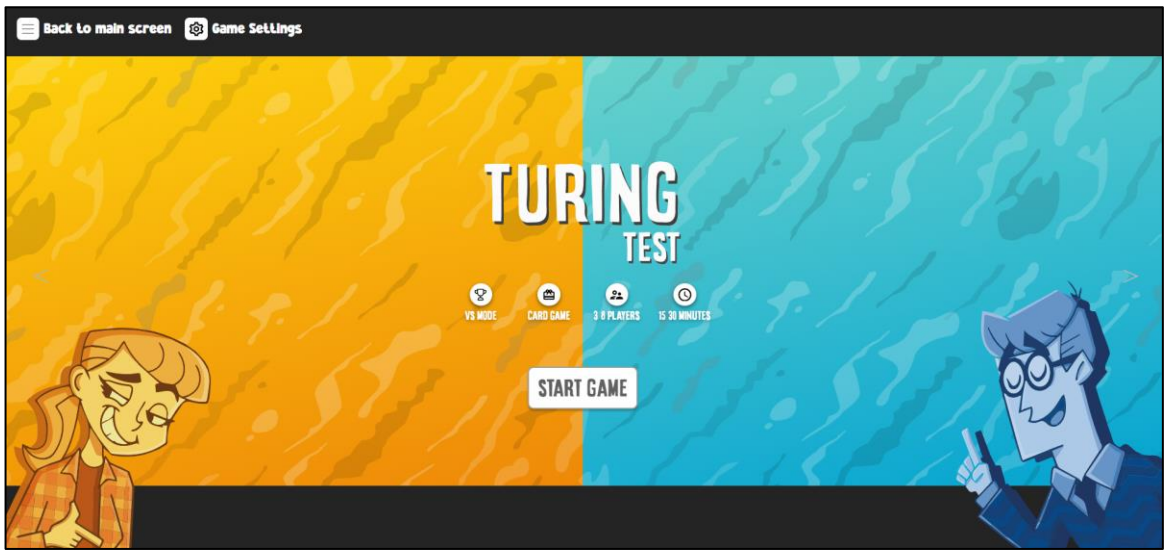


Рисунок 3.5 – Вибір гри Turing Test (знімок екрана виконано самостійно)

Третій слайд відображає гру Turing Test разом з коротким описом самої гри.

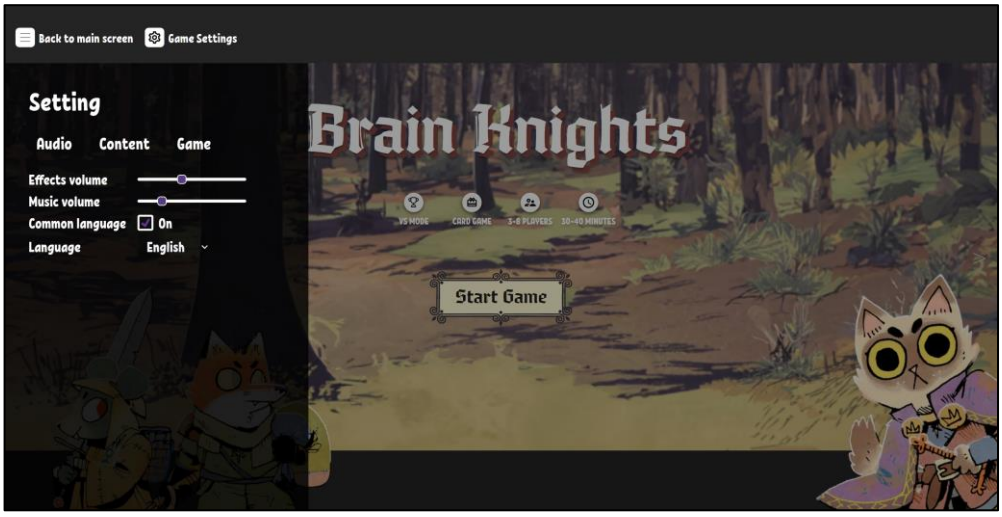


Рисунок 3.6 – Налаштуванні ігрового процесу (знімок екрана виконано самостійно)

При налаштуванні ігрового процесу користувач може змінити гучність музики та звукових ефектів, а також вибрати мову застосунку для себе або для усіх підключених контролерів.

На рисунку 3.7 показано інтерфейс ігрової кімнати Skibidy Party з підключеними гравцями, ігровим кодом для підключення та QR-кодом для безперешкодного приєднання за допомогою сканування.

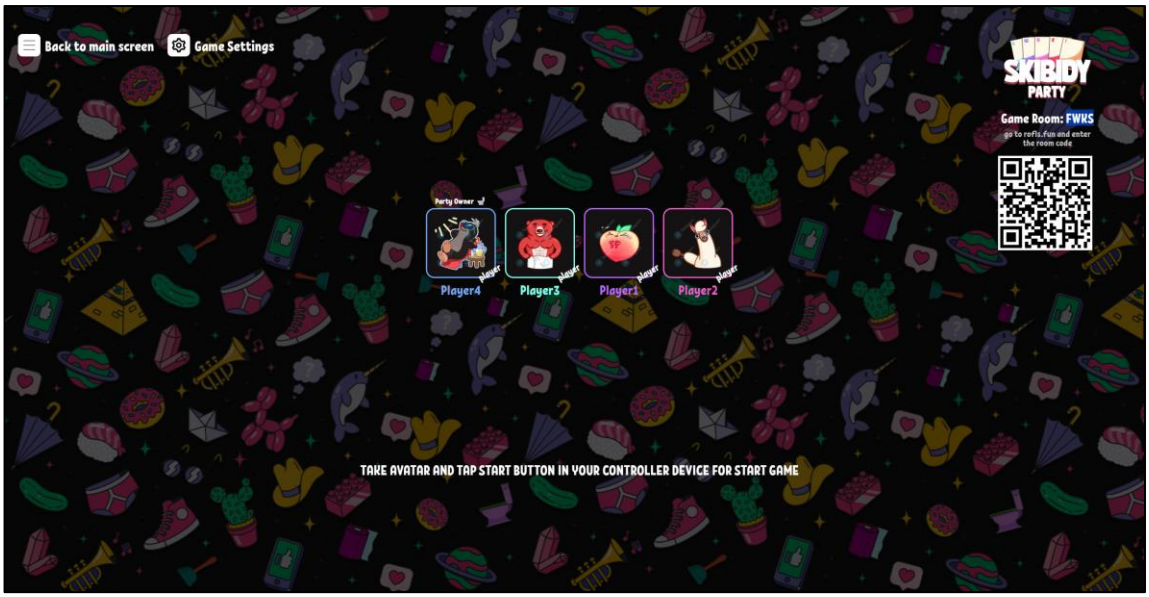


Рисунок 3.7 – Інтерфейс ігрової кімнати в грі Skibidy Party на хості (знімок екрана виконано самостійно)

На рисунку 3.8 зображене ігрове поле, де відбувається розподіл карт мемів та ситуацій перед початком раунду.

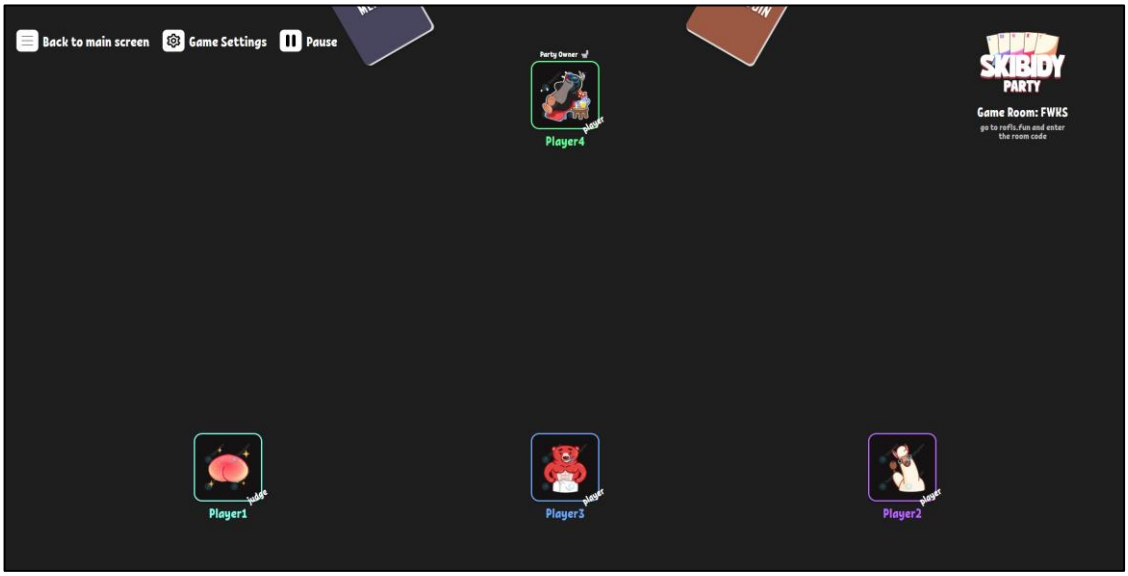


Рисунок 3.8 – Інтерфейс ігрового поля в Skibidy Party на хості (знімок екрана виконано самостійно)

На рисунку 3.9 зображена ігрова ситуація, до якої гравцям потрібно підібрати мем з набору.

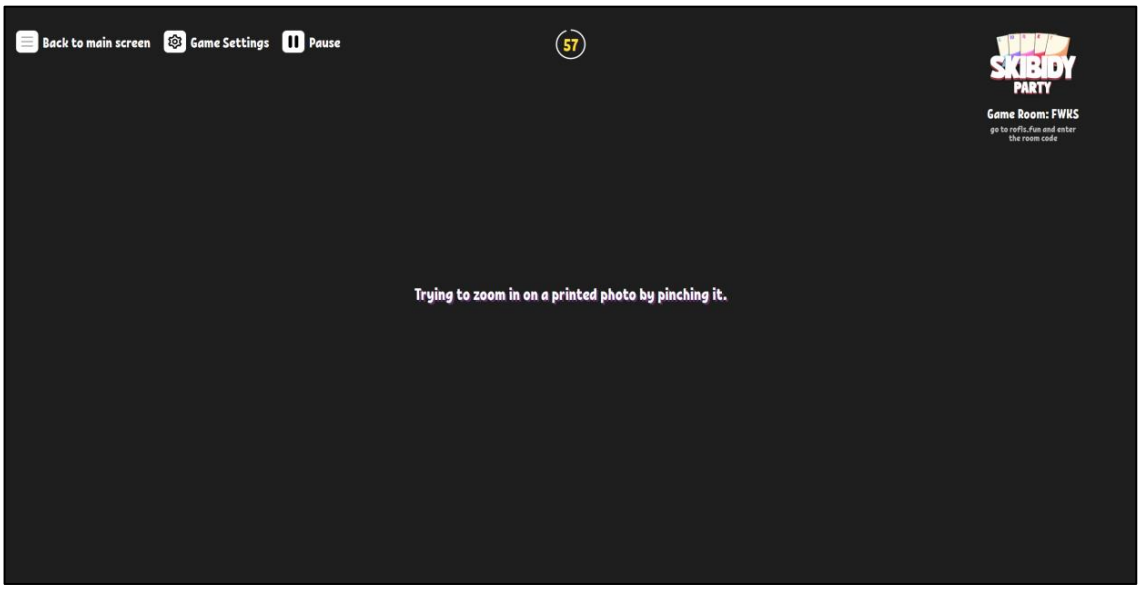


Рисунок 3.9 – Інтерфейс ігрової ситуації в Skibidy Party на хості (знімок екрана виконано самостійно)

На рисунку 3.10 зображений інтерфейс з обраними гравцями мемами. Суддя має можливість надати три призових місця для мемів.

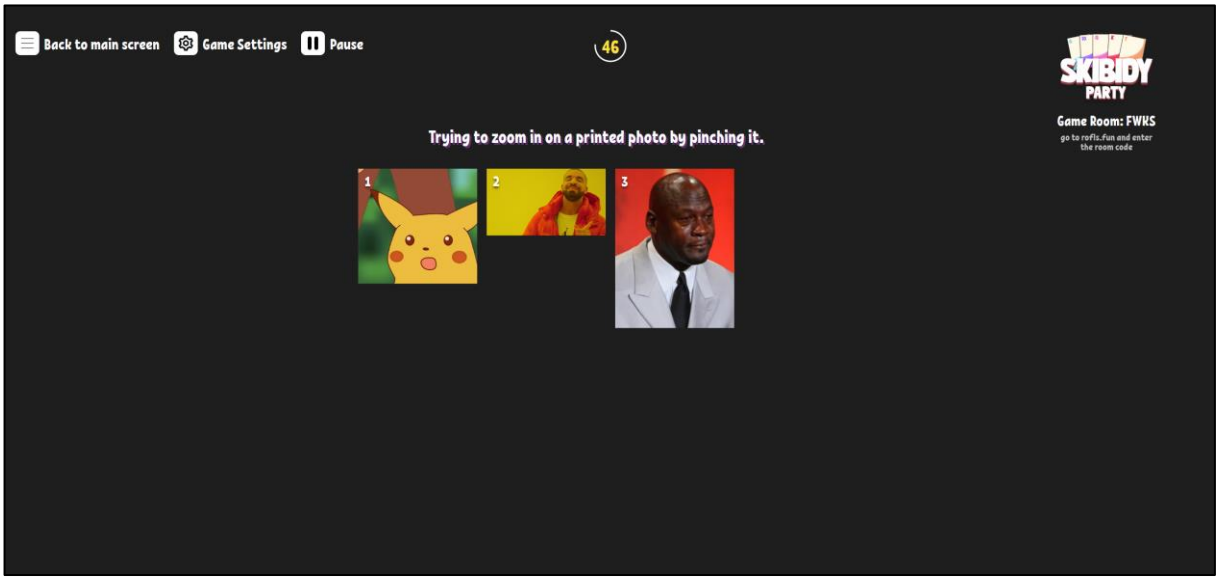


Рисунок 3.10 – Інтерфейс обраних мемів в Skibidy Party на хості (знімок екрана виконано самостійно)

На рисунку 3.11 зображений інтерфейс переможців раунду, по центру – перше місце, зліва – друге місце, справа – третє місце.

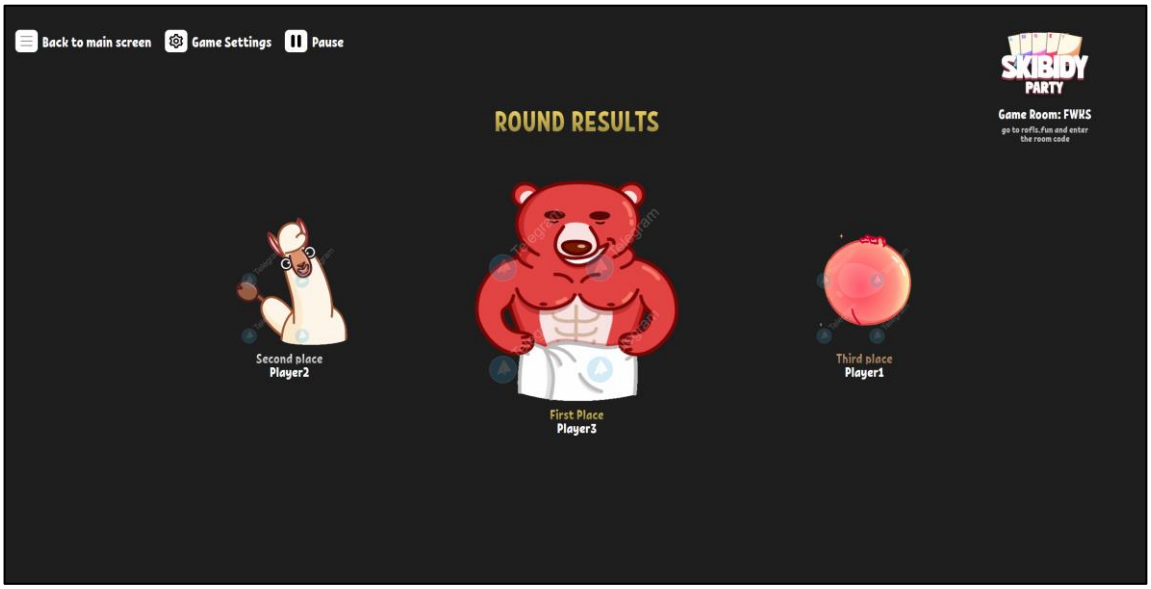


Рисунок 3.11 – Інтерфейс переможців раунду в Skibidy Party на хості (знімок екрана виконано самостійно)

На рисунку 3.12 зображена статистика очок гравців після кожного раунду, де вона динамічно оновлюється.

Рисунок 3.12 – Статистика гравців в Skibidy Party на хості (знімок екрана виконано самостійно)

На рисунку 3.13 зображений інтерфейс вибору аватарів. Також маємо дві кнопки: «Ready» відповідає за зміну статусу гравця для початку гри, «Exit» відповідає за вихід з гри.

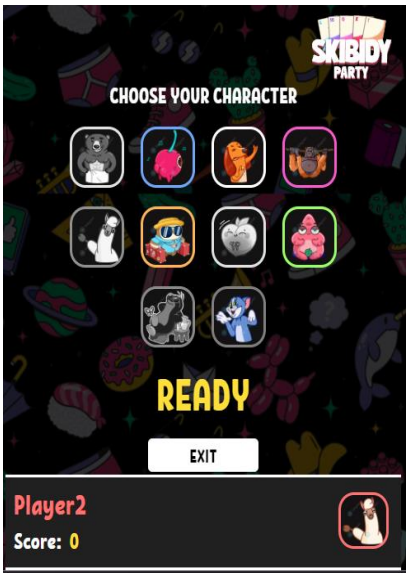


Рисунок 3.13 – Вибір аватару в Skibidy Party на контролері (знімок екрана виконано самостійно)

На рисунку 3.14 зображений інтерфейс вибору мему, де гравець обирає мем для ситуації, а потім додає його завдяки кнопці «Add meme».

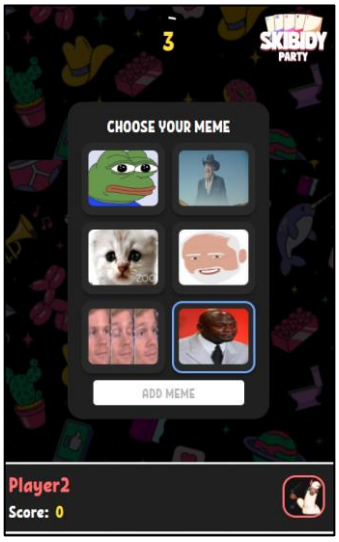


Рисунок 3.14 – Вибір мему в Skibidy Party на контролері (знімок екрана виконано самостійно)

На рисунку 3.15 зображений інтерфейс вибору переможця серед мемів суддею, Після розташування трьох призових місць, судді потрібно натиснути кнопку «Confirm».

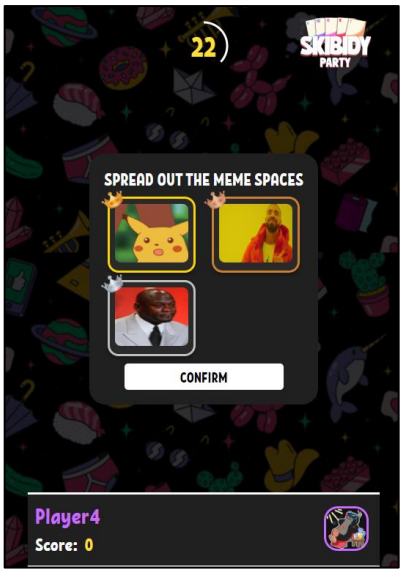


Рисунок 3.15 – Вибір переможного мему в Skibidy Party на контролері (знімок екрана виконано самостійно)

На рисунку 3.16 показано інтерфейс ігрової кімнати Turing Test з підключеними гравцями, ігровим кодом для підключення та QR-кодом для безперешкодного приєднання за допомогою сканування.

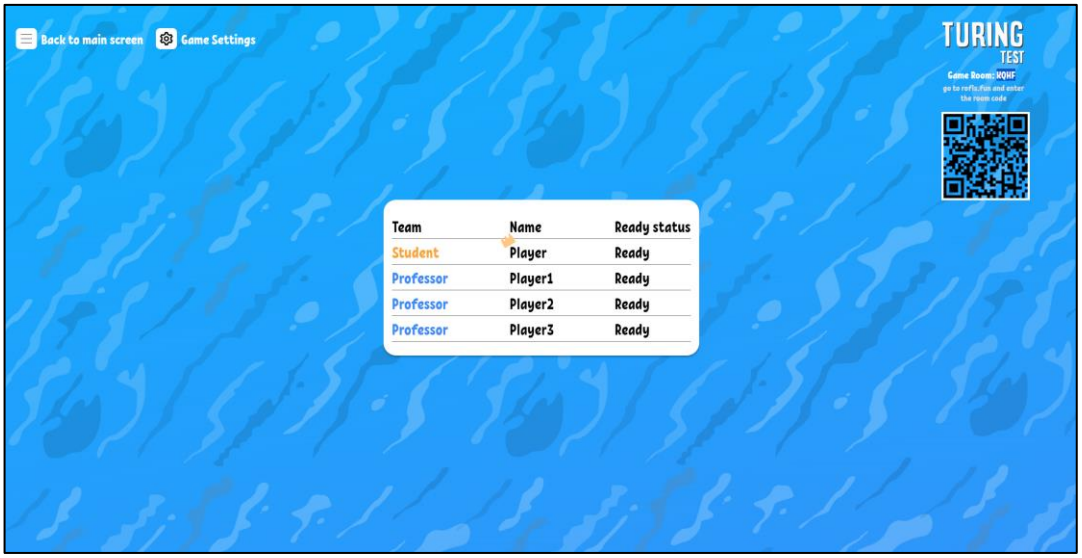


Рисунок 3.16 – Інтерфейс ігрової кімнати в грі Turing Test на хості (знімок екрана виконано самостійно)

На рисунку 3.17 зображений екран раунду, коли гравці з роллю студент, на своїх контролерах, вписують питання для професорів та ШІ.

Рисунок 3.17 – Інтерфейс раунду в грі Turing Test на хості (знімок екрана виконано самостійно)

На рисунку 3.18 зображений екран раунду, коли гравці з роллю професор, на своїх контролерах, вписують відповіді на питання студентів.

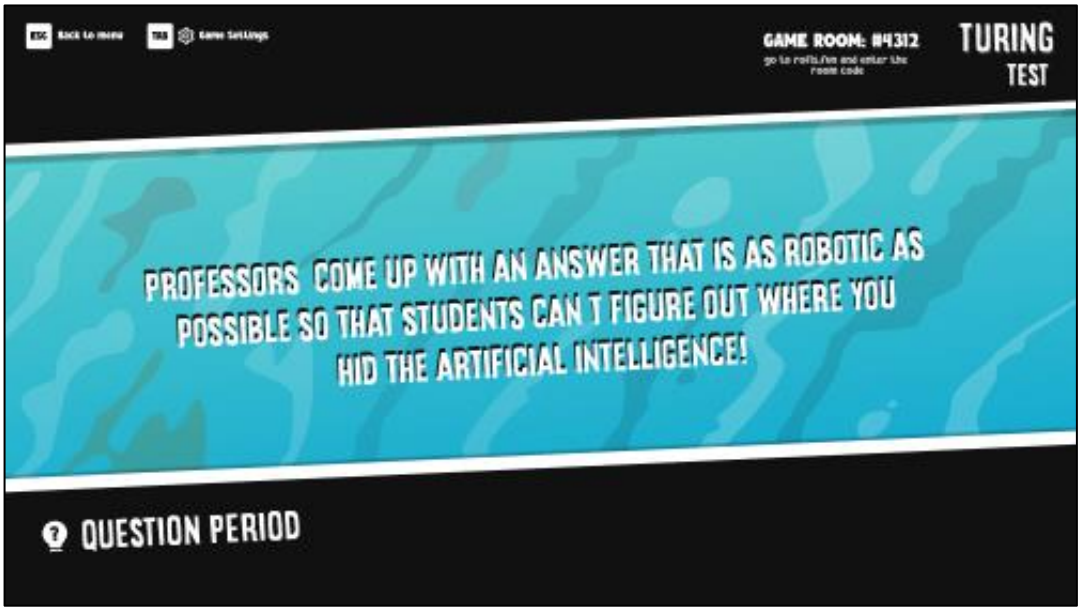


Рисунок 3.18 – Інтерфейс раунду в грі Turing Test на хості (знімок екрана виконано самостійно)

На рисунку 3.19 зображений інтерфейс варіантів відповідей професорів та ШІ на питання студентів.

Рисунок 3.19 – Інтерфейс відповідей Turing Test на хості (знімок екрана виконано самостійно)

На рисунку 3.20 зображений інтерфейс вибору відповіді. Якщо відповідь правильна – маємо синій колір, якщо неправильна – червоний.

Рисунок 3.20 – Інтерфейс вибору відповіді Turing Test на хості (знімок екрана виконано самостійно)

На рисунку 3.21 зображений інтерфейс перемоги студентів.

Рисунок 3.21 – Інтерфейс перемоги студентів в грі Turing Test на хості (знімок екрана виконано самостійно)

На рисунку 3.22 зображений інтерфейс перемоги професорів.

Рисунок 3.22 – Інтерфейс перемоги професорів в грі Turing Test на хості (знімок екрана виконано самостійно)

На рисунку 3.23 зображений інтерфейс рахунку кожної ітерації раундів та в кінці гри.

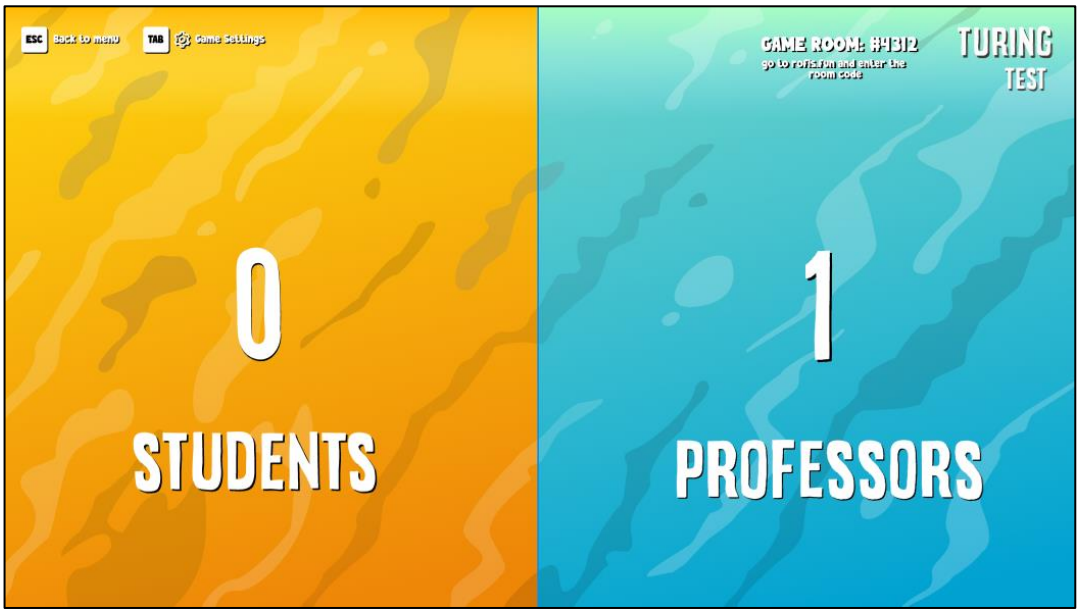


Рисунок 3.23 – Інтерфейс рахунку в грі Turing Test на хості (знімок екрана виконано самостійно)

На рисунку 3.24 зображений інтерфейс вибору команди. Також маємо три кнопки: «Ready» відповідає за зміну статусу гравця для початку гри, «Exit» відповідає за вихід з гри та «Start» для початку гри якщо користувач є власником кімнати.

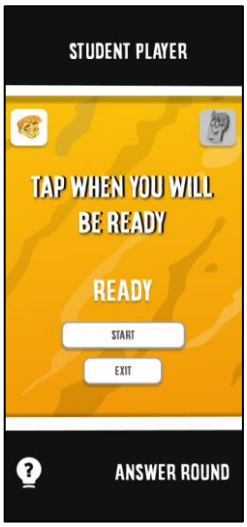


Рисунок 3.24 – Інтерфейс вибору команди в грі Turing Test на котролері (знімок екрана виконано самостійно)

На рисунку 3.25 зображений інтерфейс написання студентом питання. Для відправки питання – натиснути «Send».

Рисунок 3.25 – Інтерфейс написання питання студентами в грі Turing Test на котролері (знімок екрана виконано самостійно)

На рисунку 3.26 зображений інтерфейс написання відповіді на питання професорами. Для відправки відповіді – натиснути «Send».

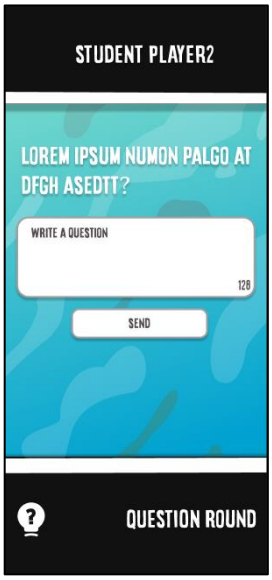


Рисунок 3.26 – Інтерфейс написання відповіді на питання професорами в грі Turing Test на котролері (знімок екрана виконано самостійно)

На рисунку 3.27 зображений інтерфейс вибору відповіді, де студенти аналізуючи, обирають відповідь ШІ. Після вибору номера відповіді для фінального голосування студенти повинні натиснути на кнопку «Send».

Рисунок 3.27 – Інтерфейс вибору відповіді ШІ в грі Turing Test на котролері (знімок екрана виконано самостійно)

На рисунку 3.28 зображений інтерфейс рахунку кожної ітерації раундів та в кінці гри.

Рисунок 3.28 – Інтерфейс в грі Turing Test на котролері (знімок екрана виконано самостійно)

На рисунку 3.29 показано інтерфейс ігрової кімнати Brain Knights з підключеними гравцями, ігровим кодом для підключення та QR-кодом для безперешкодного приєднання за допомогою сканування.

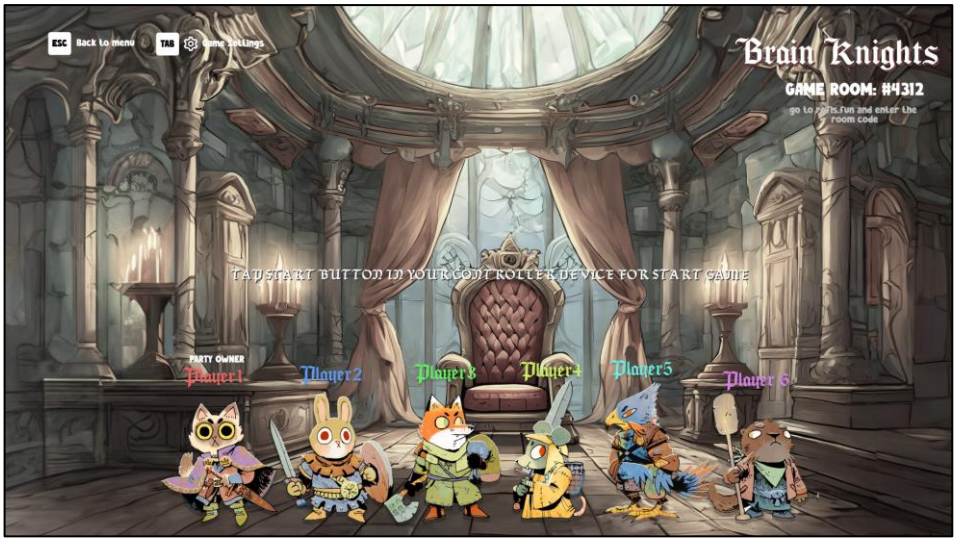


Рисунок 3.29 – Інтерфейс ігрової кімнати на головному екрані гри в грі Brain Knights на хості (знімок екрана виконано самостійно)

На рисунку 3.30 зображена тема, яку будуть розігрувати гравці в раунді.

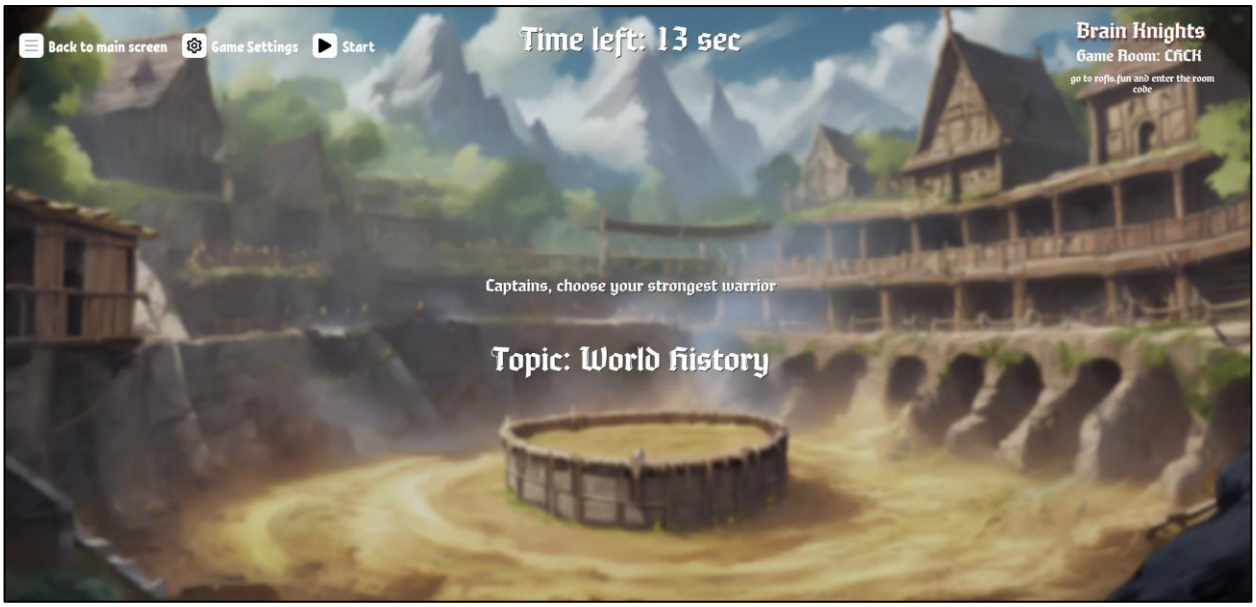


Рисунок 3.30 – Інтерфейс теми раунду в грі Brain Knights на хості (знімок екрана виконано самостійно)

На рисунку 3.31 зображено гравців які будуть відповідати на питання у бліц-раунді.

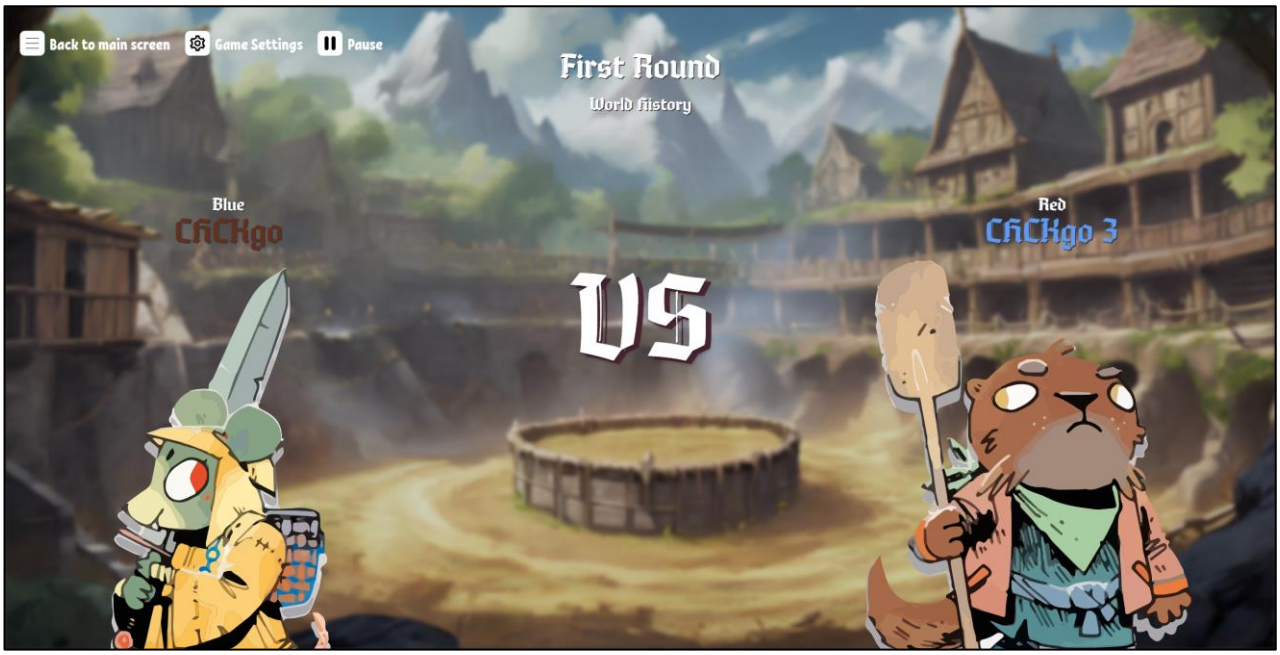


Рисунок 3.31 – Інтерфейс початку бліц-раунду в грі Brain Knights на хості (знімок екрана виконано самостійно)

На рисунку 3.32 показано бліц-раунд, де гравці відповідають швидко. У центрі екрана відображається запитання та можливі відповіді, а по боках – бали гравців.

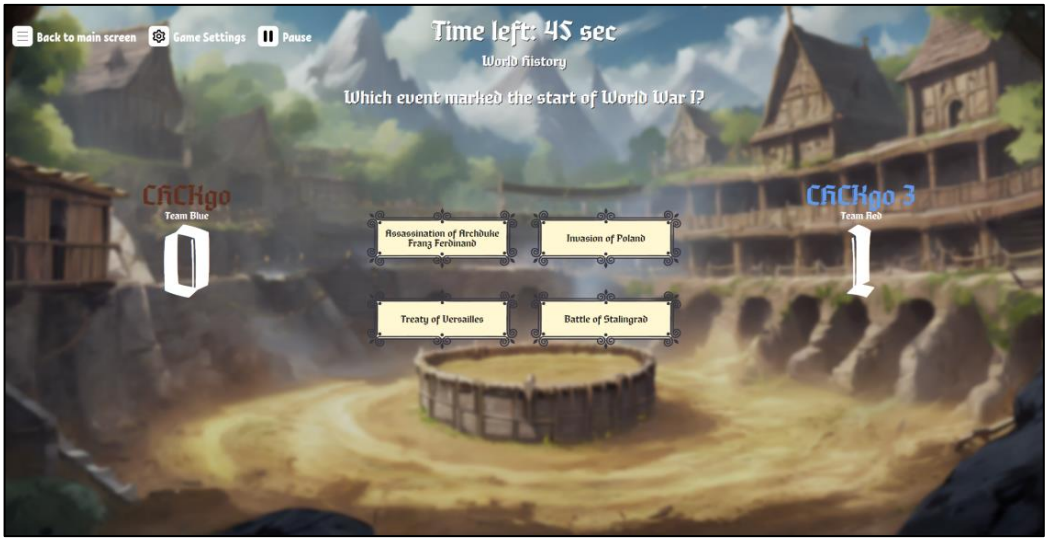


Рисунок 3.32 – Інтерфейс ігрового раунду в грі Brain Knights на хості (знімок екрана виконано самостійно)

На рисунку 3.33 зображений гравець який набрав більшу кількість балів у бліц-раунді.

Рисунок 3.33 – Інтерфейс переможця бліц-раунду в грі Brain Knights на хості (знімок екрана виконано самостійно)

На рисунку 3.34 зображений загальний рахунок після ігрового раунду. По центру екрану зображений ігровий рахунок, по бокам – команди.

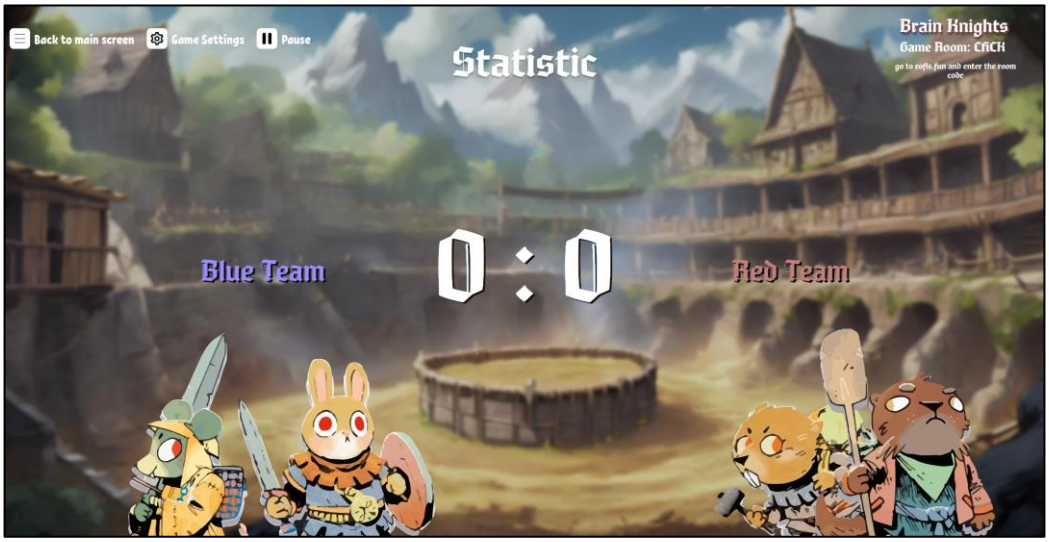


Рисунок 3.34 – Проміжні результати раунду в грі Brain Knights на хості (знімок екрана виконано самостійно)

Після того, як всі гравці завершили бліц-опитування, гра переходить до другого етапу, де гравці повинні співпрацювати, щоб відповісти на складні запитання. На рисунку 3.35 зображені теми, які капітани по черзі вилучають зі списку за допомогою контролера.

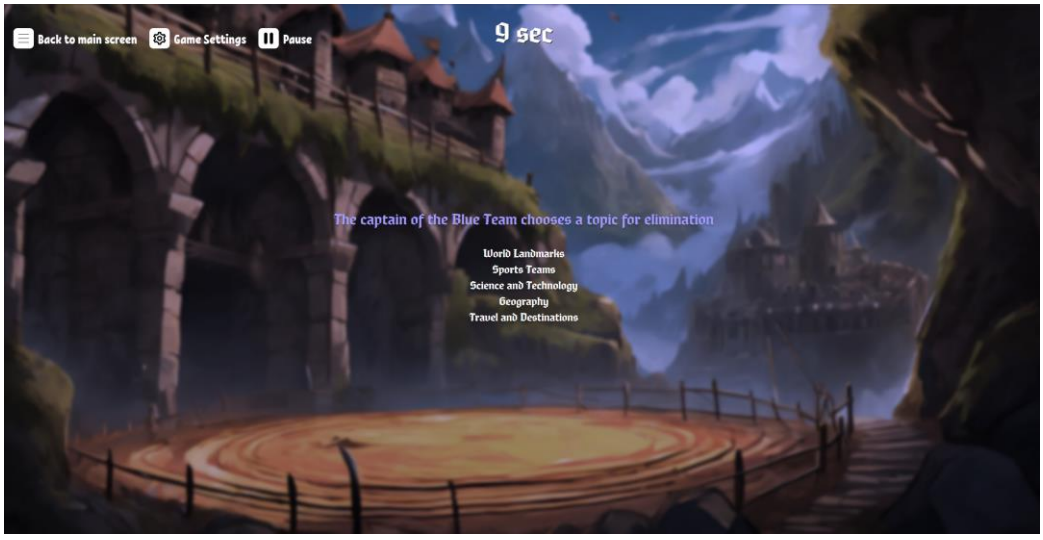


Рисунок 3.35 – Вибір теми для другої стадії в грі Brain Knights на хості (знімок екрана виконано самостійно)

На рисунку 3.36 зображений ігровий раунд у другій стадії, де гравцям потрібно командно відповідати на складні питання.

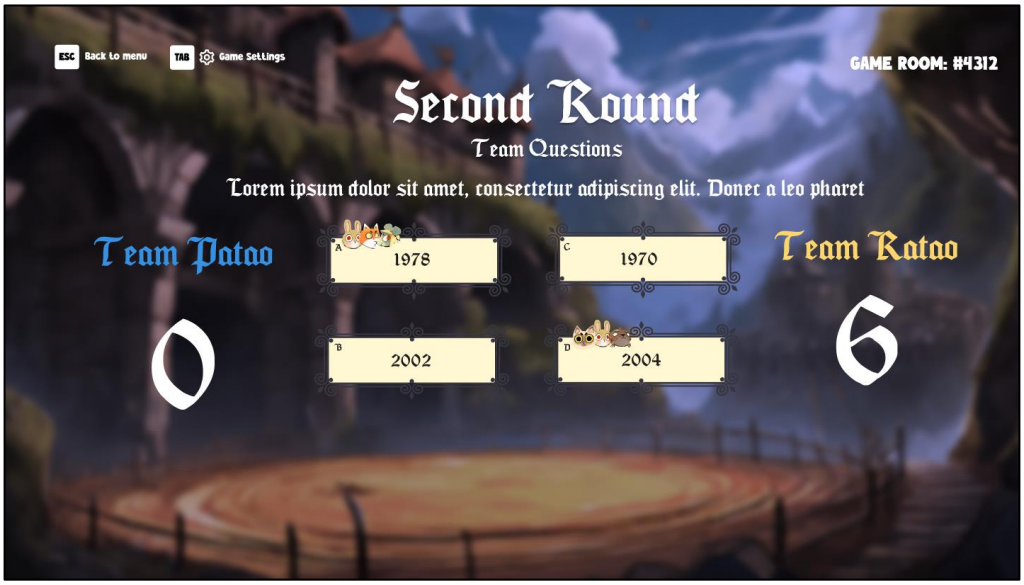


Рисунок 3.36 – Ігровий раунд у другій стадії в грі Brain Knights на хості (знімок екрана виконано самостійно)

На рисунку 3.37 зображено вибір ігрового аватару при вході до ігрової кімнати.

Рисунок 3.37 – Вибір ігрового аватара в грі Brain Knights на хості (знімок екрана виконано самостійно)

На рисунку 3.38 показано можливість для капітана команди вибрати назву команди. Крім того, гравець може змінити команду і взяти на себе роль капітана, якщо це необхідно. Як тільки всі гравці натиснуть кнопку «Ready», гра розпочнеться.

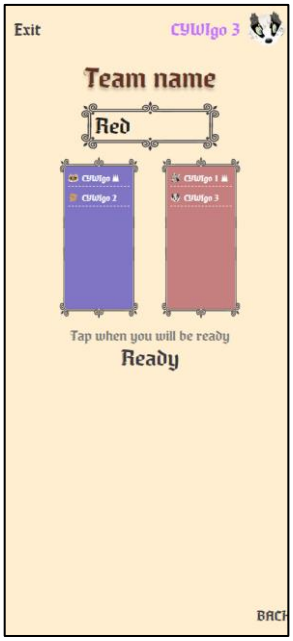


Рисунок 3.38 – Вибір команди в грі Brain Knights на хості (знімок екрана виконано самостійно)

На рисунку 3.39 зображений вибір гравця для участі у бліц-опитуванні.

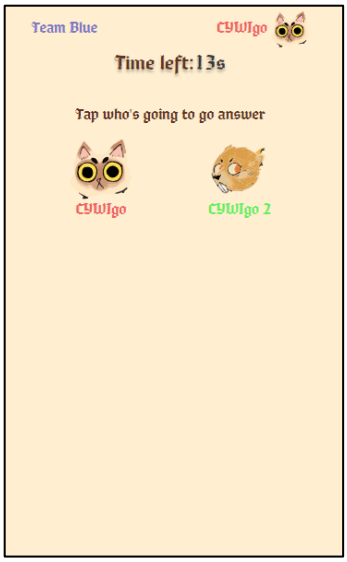


Рисунок 3.39 – Вибір гравця для відповіді в грі Brain Knights на хості (знімок екрана виконано самостійно)

На рисунку 3.40 зображений вибір відповіді гравцем на запитання, на екрані також зроблений таймер, який показує залишок часу до кінця раунду.

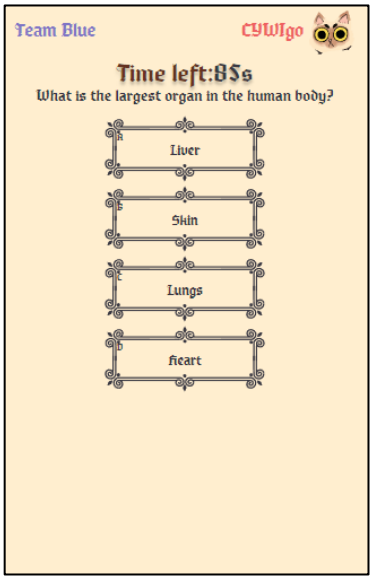


Рисунок 3.40 – Вибір відповіді на запитання в грі Brain Knights на хості (знімок екрана виконано самостійно)

Інтерфейси ретельно розроблені з дотриманням принципів зручності та простоти. Кожна гра має свій особливий стиль, що підкреслює її унікальність та висвітлює відмінності в ігровому процесі. Така увага до деталей забезпечує бездоганний користувацький досвід під час усіх взаємодій, сприяючи залученню та задоволенню гравців усіх рівнів.

3.1.2 Апаратний інтерфейс

З набору апаратних інтерфейсів необхідні базові пристрої введення/виведення, такі як клавіатура, миша, монітор та інші засоби взаємодії з користувачем та зовнішніми пристроями.

3.1.3 Програмний інтерфейс

Система інтегрує API OpenAI – модель GPT Text-Davinci-003 для імітації природної поведінки бота в міні-грі Turing Test. Крім того, система налаштована на використання HTTP та WebSocket з'єднань між сервером та клієнтом:

– HTTP з'єднання використовується для обробки запитів, які не потребують постійного обміну даними;

– WebSocket з'єднання використовується для обміну даними в реальному часі, забезпечуючи двосторонній зв'язок без необхідності його повторної ініціації.

3.1.4 Комунікаційний протокол

Для інтеграції штучного інтелекту в систему, серверна частина відправляє запит до API OpenAI. У запиті передається модель – у нашому випадку GPT Text-Davinci-003, сформований промпт на основі отриманого запитання та варіанти відповідей на нього, температура – в межах норми, для контролю результату, максимальна кількість токенів – для контролю довжини отриманої відповіді, а також у заголовку передається API ключ через bearer токен.

HTTP-запити використовуються для завантаження власного пакету мемів для ігрової сесії міні-ігри Skibidy Party, для отримання всіх існуючих кімнат та конкретної кімнати за унікальним ключем. Надіслані та отримані дані перетворюються у формат JSON.

WebSocket використовується для обробки подій у реальному часі. Це включає підключення/відключення учасників ігрової сесії, зміну мови, зміну статусу готовності гравця, зміну ігрових екранів, спробу почати ігрову сесію, зміну статусу гри, зміну аватарів, перехід в іншу команду, передачу прав капітана, зміну назви команди, вибір відповідального у раунді, видалення теми, обробку відповіді

на питання, зміну обраного мему, відправлення обраного мему, вибір кращого мему, розподіл місць переможців, відправлення питання та відповіді, а також вибір певної відповіді. Надіслані та отримані дані перетворюються у формат JSON.

3.1.5 Обмеження пам’яті

Для оптимізації роботи програми та запобігання витоку пам’яті необхідно регулярно перевіряти та оптимізувати використання ресурсів. Використання інструментів моніторингу та профілювання коду може допомогти виявити та усунути проблеми з витоком пам’яті. Деякі корисні підходи включають уникання надмірного створення об’єктів та ефективне використання можливостей збору сміття.

Враховуючи технічні вимоги, гра повинна функціонувати оптимально з використанням не більше 500 МБ оперативної пам’яті. Максимальний розмір завантажуваних текстур не повинен перевищувати 10 МБ. Гра розроблена з урахуванням пристроїв, які мають не менше 4 ГБ оперативної пам’яті.

3.1.6 Операції

На стороні хоста:

– користувач може перейти до вибору однієї з наявних міні-ігор з головного меню після натискання кнопки «Start»;

– на слайдері з міні-іграми користувач може обрати одну з запропонованих ігор, переглянути інформацію про неї та натиснути кнопку «Start», щоб відправити запит на сервер для створення приватної ігрової кімнати;

– після початку ігрової сесії користувач може переглядати загальну інформацію, що відображається на екрані та змінюється відповідно до логіки гри, яка обробляється на сервері;

– під час розпочатої ігрової сесії користувач може змінити статус гри: призупинити гру або продовжити після паузи, натиснувши кнопку «Pause» або «Play», а також повернутися до головного меню, натиснувши кнопку «Back to main screen»;

– у меню та під час ігрової сесії користувач може налаштувати ігрову сесію: змінити гучність музики та ефектів, а також змінити мову або лише на хості, або на хості та на усіх підключених контролерах до кімнати.

На стороні контролера:

– гравець може приєднатися до приватної ігрової кімнати, ввівши свій нікнейм та ключ кімнати, що відображається на хості. Якщо ключ введено правильно, з’явиться кнопка з назвою гри, при натисканні на яку він приєднується до неї;

– усі гравці можуть змінити свій статус готовності, натиснувши кнопку «Ready» або «Not ready». Якщо гравець – власник кімнати, він може розпочати ігрову сесію після того, як набереться мінімальна кількість гравців і всі вони підтвердять свій статус готовності;

– якщо під час розпочатої ігрової сесії гравець відключився від кімнати, він може ввести ключ кімнати на головному екрані контролера та перепідключитися до неї;

– у меню та під час ігрової сесії гравець може змінити мову застосунку;

– у міні-грі Brain Knights перед початком гри гравець може обрати свій аватар та змінити команду. Якщо гравець – капітан команди, він може змінити назву команди, а також передати права капітанства іншому учаснику своєї команди;

– у міні-грі Brain Knights капітани команд на етапі вибору відповідального на певну тему можуть обрати гравця зі своєї команди, який буде відповідати на запитання;

– у міні-грі Brain Knights гравець, якого обрали для відповіді на запитання на першій стадії гри, може обрати одну з запропонованих відповідей на кожне виведене питання;

– у міні-грі Brain Knights капітани команд на етапі виключення тем можуть обрати почергово одну з тем зі списку, щоб видалити її;

– у міні-грі Brain Knights капітани команд на етапі відповіді на другій стадії гри можуть обрати одну з запропонованих відповідей на кожне виведене питання та обговорити його разом зі своєю командою;

– у міні-грі Skibidy Party перед початком гри гравець може обрати свій аватар;

– у міні-грі Skibidy Party звичайні гравці на етапі виведення ситуації можуть обрати одну з карток мемів, що опинилася у них на руках;

– у міні-грі Skibidy Party гравець, якому випала роль судді на етапі розподілення місць, може розподілити призові місця гравців від першого до третього на власну думку;

– у міні-грі Turing Test перед початком гри гравець може змінити свою команду;

– у міні-грі Turing Test гравці команди студентів на етапі написання запитання для команди професорів можуть ввести своє запитання та відправити його;

– у міні-грі Turing Test гравці команди професорів на етапі написання відповідей на запитання команди студентів можуть написати свої відповіді на отримані запитання та відправити їх;

– у міні-грі Turing Test гравці команди студентів на етапі вибору відповіді бота можуть обрати одну з відповідей зі списку, яка, на їхню думку, належить боту, та проголосувати за неї.

Операції розподіляються між хостом і контролерами: хост керує ігровими сесіями, а контролери дозволяють гравцям приєднуватися, змінювати статуси готовності та виконувати дії в міні-іграх. Кожна міні-гра має унікальні етапи та можливості для гравців, забезпечуючи різноманітність геймплею.

3.1.7 Функції продукту

На хості:

а) загальне:

1) забезпечення зручної навігації зі збереженням інформації про останню обрану міні-гру;

2) вибір однієї з наявних міні-ігор на головному екрані (Brain Knights, Turing Test або Skibidy Party);

3) створення приватної ігрової кімнати для кожної ігрової сесії;

4) управління поточним станом розпочатої ігрової сесії;

5) відображення інформації про створену приватну кімнату для можливості підключення до неї;

6) відображення часу, що залишився до зміни етапу гри;

7) відображення гравця, який є власником ігрової сесії;

8) відображення нікнеймів, стану підключення та готовності кожного з гравців;

9) зміна поточного екрану відповідно до даних, що були отримані з сервера;

10) валідація даних, що відправляються на сервер;

б) міні-гра Brain Knights:

1) відображення приналежності кожного гравця до певної команди та їх капітанів;

2) відображення аватарів гравців та назв команд;

3) відображення кількості правильних відповідей, які надав кожен гравець;

4) відображення теми вікторини та її запитань;

5) відображення правильних та неправильних відповідей;

6) налаштування логіки відображення правильних відповідей гравців (на першому етапі вони виводяться одразу після відповіді одного з учасників, а на другому – після відповідей усіх капітанів команд);

7) відображення аватарів гравців або команди поруч із обраним варіантом відповіді;

8) відображення переможця раунду;

9) відображення загального рахунку команди;

10) відображення MVP-гравця першого етапу гри;

11) відображення тем для виключення та інформації про черговість;

12) відображення результату гри;

13) відображення титулів гравців у кінці гри;

в) міні-гра Turing Test:

1) відображення приналежності кожного гравця до певної команди;

2) відображення інструкцій щодо виконання завдань;

3) відображення поставлених запитань та наданих відповідей;

4) відображення наданих голосів за варіанти відповідей, які, на думку студентів, надав бот;

5) відображення відповіді, яку надав бот, а також результату голосування команди студентів;

6) відображення переможців та проміжних результатів раундів;

7) відображення результату гри;

8) відображення титулів гравців у кінці гри;

г) міні-гра Skibidy Party:

1) завантаження власного набору мемів до ігрової сесії;

2) валідація завантажуваних файлів;

3) відображення аватарів та ролей гравців;

4) відображення обраної ситуації;

5) відображення мемів, які обрали гравці;

6) відображення переможців кожного раунду;

7) відображення очок, які заробив кожен гравець за раунд;

8) відображення мемів, які обрали гравці;

9) відображення результату гри;

10) відображення титулів гравців у кінці гри.

На контролері:

а) загальне:

1) зміна локалізації застосунку;

2) відключення та зміна гучності звуку;

3) можливість вписати нікнейм гравця;

4) можливість під’єднатися до ігрового лобі за допомогою унікального сгенерованого коду нікнейм гравця;

5) можливість перепідключення до лобі за допомогою унікального коду;

6) можливість поставити гру на паузу у будь-який момент гри;

7) відображення гравця який створив лобі;

б) міні-гра Brain Knights:

1) можливість обрати унікальний аватар для гравця;

2) можливість змінити команду;

3) можливість змінити капітана команди;

4) можливість обрати назву для команди якщо гравець грає у ролі капітана;

5) можливість капітану обрати гравця який буде відповідати у бліц-раунді;

6) можливість вибрати одну з чотирьох відповідей;

7) можливість обрати тему для раунда методом виключення;

8) перегляд інформації про користувача;

9) перегляд стану паузи;

в) міні-гра Turing Test:

1) можливість змінити команду;

2) можливість вписати питання для штучного інтелекту, якщо гравець у команді студентів;

3) можливість вписати відповідь на питання якщо гравець у команді професорів;

4) можливість проголосувати за найпідозрілішу, на думку гравця, відповідь;

5) відображення результату гри;

6) відображення рахунку;

г) міні-гра Skibidy Party:

1) можливість відзначити готовність та почати гру;

2) обрати ігрову карту з переліку розданих;

3) оцінити ігрові карти та обрати призові місця;

4) відображення підрахованого рахунку та статистики;

5) відображення стану паузи.

На сервері:

а) загальне:

1) обробка всіх дій гравців, які передбачені геймплеєм та передбачають синхронізацію з серверною частиною;

2) обробку дій гравців, які не передбачені геймплеєм, але можуть бути викликані через середовище, в якому працює гра;

3) ідентифікація гравців;

4) створення і управління гральними кімнатами;

5) синхронізація гри між усіма гравцями;

6) обробку та збереження даних сесії гри протягом часу їх існування;

7) завантаження і управління файлами зображень;

8) обробка статичних текстових даних, необхідних для роботи ігор, таких як питання, відповіді на них та опис ситуацій, які зберігаються українській та англійській мовах;

б) для міні-ігор Brain Knights, Turing Test та Skibidy Party:

1) організатор повинен мати можливість створити та увійти в кімнату;

2) гравець повинен мати можливість увійти в кімнату;

3) система повинна привласнити гравцю, який увійшов першим, статус власника кімнати;

4) гравець повинен мати можливість вийти з кімнати;

5) якщо власник кімнати вийшов, система повинна автоматично передати статус власника кімнати наступному в черзі гравцю, який увійшов після попереднього власника кімнати;

6) організатор повинен мати можливість змінити мову в кімнаті;

7) організатор повинен мати можливість встановити мову кімнати загальною для всіх її учасників;

8) власник кімнати повинен мати можливість почати ігрову сесію для учасників кімнати;

9) організатор повинен мати можливість ставити ігрову сесію на паузу та відновлювати її роботу;

10) організатор повинен мати можливість ставити ігрову сесію на паузу та відновлювати її роботу;

11) організатор повинен мати можливість вийти з кімнати;

12) якщо організатор вийшов, то система повинна завершити ігрову сесію та очистити її дані;

13) користувачі повинні мати можливість отримувати дані про кімнату, ігрову сесію в ній та інформацію про себе;

14) серед гравців повинні бути розподілені досягнення, які вони здобули протягом гри;

15) система повинна перед початком гри формувати чергу відображення екранів гри для кожного типу гравця; всі елементи черги повинні містити множину елементів, при цьому кожен елемент цієї множини повинен включати назву екрану, інформацію про те, для кого він призначений, тривалість його відображення та функції, які необхідно виконати під час, перед або після його появи; система повинна мати можливість змінювати чергу екранів та її елементи, а також ті елементи, які вже вийшли з неї, при необхідностях, що випливають з логіки гри;

16) система повинна виконувати валідацію отриманих даних;

17) користувачі повинні отримувати інформацію про час відображення поточного екрану;

в) для міні-гри Brain Knights:

1) після того, як гравець увійшов в кімнату, де ігрова сесія ще не розпочалася, серверна частина повинна автоматично надати йому випадковий аватар;

2) повинна надавати можливість гравцям змінити аватар;

3) повинна надавати можливість гравцям змінити команду;

4) повинна розподіляти гравців по командам, де менше гравців; якщо гравців немає або їх рівна кількість, то повинна помістити гравця в червону команду;

5) повинна надавати роль капітана першим гравцям у командах;

6) повинна видаляти роль капітана у гравця, який перейшов в іншу команду, де вже є капітан;

7) повинна надавати можливість капітану передати цю роль іншому гравцю;

8) повинна надавати можливість користувачам отримувати інформацію про команди та гравців у них;

9) повинна визначати кількість раундів;

10) повинна випадковим чином обирати тему для питань на час одного раунду;

11) повинна надавати можливість капітану обрати гравця, який буде відповідати на запитання з обраної теми;

12) повинна надавати можливість користувачам інформацію про тему, випадкове питання та варіанти відповіді до нього;

13) повинна надавати можливість організатору отримати кількість правильних відповідей, які надав кожен гравець;

14) повинна надавати можливість організатору отримати інформацію про те, яку відповідь обрав гравець і чи вона є правильною або неправильною;

15) повинна виконувати підрахунок балів для кожної команди в кінці кожного раунду;

16) повинна надати можливість організатору отримати інформацію про загальний рахунок балів для кожної команди;

17) повинна надавати можливість капітанам під час другого етапу гри обирати теми, які необхідно вилучити;

18) повинна визначати команду, яку перемогла у грі;

19) повинна рахувати кількість правильних відповідей кожного гравця;

20) повинна визначати найціннішого гравця гри – того, хто надав найбільшу кількість правильних відповідей;

21) повинна надавати можливість капітану змінити назву команди;

22) повинна надавати можливість гравцям обирати відповіді на запитання;

г) для міні-гри Turing Test:

1) повинна розподіляти гравців на команди студентів та професорів у такому співвідношенні: чотири гравці – три професори і один студент; п’ять гравців – три професори і два студенти; шість гравців – чотири професори і два студенти; сім гравців – чотири професори і три студенти; вісім гравців – п’ять професорів і три студенти;

2) повинна надавати можливість гравцям змінити команду відповідно до правил, якщо є вільні місця;

3) повинна надавати можливість гравцям з команди студентів надсилати питання;

4) повинна надавати можливість гравцям з команди професорів отримувати питання;

5) повинна надавати можливість гравцям з команди професорів надсилати відповідь на питання;

6) повинна мати налаштовану логіку штучного інтелекту, який за допомогою зазначених налаштувань мав би надавати природні відповіді. У випадку помилки, сервер повинен повертати випадкову відповідь з заздалегідь підготовленого списку;

7) повинна надавати можливість організатору отримувати до кожного питання всі відповіді, як від професорів, так і від штучного інтелекту, без вказання, чия це відповідь;

8) повинна надавати можливість гравцям з команди студентів надсилати свої вибори відповідей, які, на їх думку, надав штучний інтелект;

9) повинна надавати можливість організатору отримувати відповіді, які обирають гравці з команди студентів;

10) повинна обробляти відповіді, які обрали гравці з команди студентів;

11) повинна надавати можливість організатору отримувати дані про те, чи помилилися гравці з команди студентів під час вибору штучного інтелекту чи ні;

12) повинна нараховувати бали команді переможців;

13) повинна надавати можливість користувачам отримувати дані про нараховані бали, а також дані про переможця, як в кінці кожного раунду так і в кінці гри;

д) для міні-гри Skibidy Party:

1) повинна надавати можливість організатору завантажувати власні файли зображень мемів;

2) повинна виконувати валідацію файлів, які завантажуються;

3) повинна створювати колоду карток з урахуванням завантажених мемів або використовуючи лише системні;

4) після того, як гравець увійшов в кімнату, де ігрова сесія ще не розпочалася, серверна частина повинна автоматично надати йому випадковий аватар;

5) повинна надавати можливість гравцям змінити аватар;

6) повинна перед початком кожного раунду надати випадковому гравцеві, який ще не був суддею, роль судді, і іншим – роль звичайного гравця;

7) повинна надавати можливість користувачам отримувати файли зображень мемів;

8) повинна обирати випадковим чином ситуацію, яка ще не була обрана раніше;

9) повинна надавати можливість організатору отримати випадково обрану ситуацію для раунду;

10) повинна розподіляти картки між звичайними гравцями, з виключенням тих карток, які вже були розподілені;

11) повинна надавати можливість звичайним гравцям надсилати свій вибір найсмішнішого мема;

12) повинна надавати можливість суддям отримувати інформацію про вибір найсмішніших мемів за думкою звичайних гравців;

13) повинна надавати можливість суддям надсилати дані про призначені мемам місця від першого до третього;

14) повинна нараховувати бали звичайним гравцям в залежності від місця, на яке поставив їх мем суддя;

15) повинна надавати можливість організаторам отримувати інформацію про бали гравців з попереднього раунду та поточного;

16) повинна надавати можливість користувачам отримувати дані про переможців раунду та гри.

Надавши широкий спектр ігрових можливостей та інтерактивних функцій для користувачів, продукт забезпечує захоплюючий геймплей та можливість спільного відпочинку для гравців усіх рівнів.

3.1.8 Припущення та залежності

Основні функції, які будуть включені в початковий випуск продукту, включають наступне:

– додаток буде запускатися в сучасних веб-браузерах, таких як Google Chrome, Mozilla Firefox, Safari, Yandex, Opera тощо;

– для використання системи зі сторони хоста необхідний доступ до комп’ютера або консолі з веб-браузером та підключенням до Інтернету;

– для використання системи зі сторони контролера необхідний доступ до смартфона або іншого пристрою з веб-браузером та підключенням до Інтернету;

– додаток може бути використаний громадянами будь-якої країни;

– користувачі матимуть змогу обирати мову інтерфейсу – українську або англійську – на своєму пристрої;

– додаток матиме інтеграцію штучного інтелекту на серверній частині моделі GPT Text-Davinci-003 для покращення геймплею;

– збереження даних ігрових сесій відбуватиметься у пам’яті серверу;

– використання ігрового програмного застосунку дозволить компанії друзів та знайомих обрати одну з доступних міні-ігор на хості та підключитися до ігрової сесії за допомогою власних девайсів для зручного та ефективного керування ігровим процесом.

3.2 Властивості програмного продукту

Зважаючи на вимоги до програмного продукту, необхідно врахувати наступне:

а) масштабованість:

1) архітектура програми повинна бути гнучкою та розширюваною, щоб забезпечити можливість додавання нових ігрових режимів або функцій у майбутньому без значних змін у коді;

2) застосунок має мати можливість легко масштабуватися для підтримки великої кількості одночасних ігрових сесій і гравців;

б) продуктивність:

1) програма повинна забезпечувати плавний ігровий процес без значних затримок або зависань, навіть при підключенні кількох гравців одночасно;

2) час завантаження ігрових сесій та переходів між екранами має бути мінімальним;

в) сумісність:

1) система повинна бути сумісною з різними веб-браузерами та пристроями для широкого охоплення аудиторії;

г) керованість і зручність використання:

1) програмний продукт має інтуїтивний і легкий у використанні інтерфейс користувача, що дозволяє гравцям взаємодіяти з системою без зайвих труднощів;

2) додаток розбитий на дві частини – хост та контролер, що дозволяє охопити велику кількість користувачів та забезпечити проведення ігрових сесій у будь-якому місці за наявністю пристроїв, які мають браузер та доступ до Інтернету;

д) оновлення та підтримка:

1) після випуску першої версії продукту планується проводити регулярні оновлення контенту ігор, додавати нові функції та створювати нові міні-ігри;

е) локалізація:

1) інтерфейс програмного застосунку повинен бути реалізований українською та англійською мовами;

2) зміна локалізації має відбуватися не лише для програмного застосунку, але й для всіх підключених контролерів до ігрової сесії за потреби;

ж) надійність:

1) система повинна забезпечувати безперебійний доступ для користувачів, навіть у випадку непередбачених ситуацій або помилок;

и) мережеве налаштування:

1) система повинна мати налаштоване WebSocket-з’єднання між клієнтською та серверною сторонами;

2) система повинна забезпечити можливість розірвати WebSocket з’єднання з клієнтської сторони;

3) система повинна забезпечити обмін даними між клієнтською та серверною сторонами через WebSocket-з’єднання.

Після переліку ключових характеристик програмного продукту відзначається важливість забезпечення якості та функціональності для задоволення потреб користувачів. Постійна увага до вдосконалення, розвитку та підтримки продукту є запорукою успішної ігрової платформи. Надійність, продуктивність та зручність використання є основними пріоритетами у впровадженні ігрових інновацій, які надихають та залучають широку аудиторію гравців.

3.3 Атрибути програмного продукту

3.3.1 Надійність

Система повинна мати високу надійність для забезпечення безперебійної роботи та задоволення потреб користувачів. Для досягнення цієї мети враховуються наступні аспекти: доступність сервісу, стійкість до помилок, відновлення після збоїв, моніторинг та логування.

3.3.2 Доступність

Програмний застосунок повинен бути доступним для користувачів з різними рівнями ігрового досвіду. Забезпечення простих та зрозумілих механік, розроблених для міні-ігор, дозволить новачкам швидко опанувати гру. Також, потрібно врахувати можливості адаптивного дизайну, щоб програма коректно відображалася на різних типах пристроїв, включаючи консолі, комп’ютери, планшети та мобільні телефони.

3.3.3 Безпека

З’єднання між сервером і клієнтом буде забезпечено за допомогою протоколу HTTPS, що забезпечить шифрування даних та забезпечить безпеку передачі інформації між ними. Регулярні аудити та вдосконалення системи захисту допоможуть запобігти можливим загрозам і забезпечити безпеку під час гри.

3.3.4 Супроводжуваність

Забезпечення постійної підтримки та вчасного реагування на запити користувачів є ключовим аспектом. Плановані випуски оновлень та виправлення помилок дозволять забезпечити стабільну роботу програми та враховувати потреби користувачів.

3.3.5 Переносимість

Застосунок має працювати на різних типах пристроїв, включаючи консолі, комп’ютери, планшети та мобільні телефони, забезпечуючи зручний доступ до гри в будь-який час і в будь-якому місці. Також важливо забезпечити сумісність з основними веб-браузерами та пристроями для безперебійної роботи на різних типах обладнання.

3.3.6 Продуктивність

Програма повинна забезпечувати плавний ігровий процес без значних затримок або зависань, навіть при підключенні кількох гравців одночасно. Час завантаження ігрових сесій та переходів між екранами має бути мінімальним. Система повинна ефективно використовувати ресурси пристрою, на якому виконується, для забезпечення високої продуктивності та стабільної роботи.

3.4 Вимоги бази даних

У цьому проєкті не передбачено використання бази даних. Замість цього, дані ігрових сесій будуть зберігатися безпосередньо в пам’яті сервера.

3.5. Інші вимоги

ДОДАТОК Г

Тест-план ігрового програмного застосунку у жанрі multiplayer party games

---

EXLEVEN

ROFLSFUN

Тест-план

1.0

Історія версій

Дата	Опис	Автор	Коментарі
15.05.2024	Версія 1.0	Команда проєкту ROFLSFUN	Перша редакція

Затвердження документів

Наступний тест-план був прийнятий та схвалений наступними особами:

Підпис	Друковане ім’я	Заголовок	Дата
	О. Ю. Калініченко	Back-end частина, інтеграція AI до системи	15.05.2024
	В. О. Бухало	Back-end частина, налаштування мережевої гри	15.05.2024
	А. О. Двугрошев	Хост частина, левел-дизайн та UI/UX	15.05.2024
	Є. В. Мірошніков	Контролер, баланс ігрового процесу	15.05.2024

ЗМІСТ

1 ВСТУП

1.1 Мета

Цей документ тест-плану для ігрового програмного застосунку у жанрі multiplayer party games має наступні цілі:

– визначити наявну інформацію про проєкт та програмні компоненти, які необхідно протестувати;

– перерахувати рекомендовані вимоги до тестування на високому рівні;

– порекомендувати та описати стратегії тестування, які будуть використані;

– визначити необхідні ресурси та надати оцінку зусиль, необхідних для тестування;

– перелічити елементи, які повинні бути отримані в результаті тестового проєкту.

1.2 Передумови

Об’єктом тестування є ігровий програмний застосунок у жанрі multiplayer party games, призначений для організації веселих та інтерактивних онлайн-зустрічей з друзями. Основна мета цього застосунку – забезпечити гравців набором простих, але захоплюючих ігор, які сприяють соціальній взаємодії та розвагам. Ігровий застосунок включає три основні типи ігор: гумористичну карткову гру, соціальну дедуктивну гру та змагальну вікторину.

Основні функції та можливості застосунку охоплюють керування ігровими сесіями, гравцями, ігровою логікою та синхронізацію дій між учасниками. Гравці можуть використовувати свої смартфони, планшети або інші цифрові пристрої з доступом до Інтернету для керування ігровим процесом, тоді як трансляція ігрового процесу здійснюється на великому екрані для зручності спостереження.

Гравці, які транслюють ігровий процес, також мають можливість налаштовувати звуковий супровід і локалізацію гри.

Клієнтська частина застосунку базується на класичній архітектурі односторінкової програми, що ґрунтується на розділенні функціональності на три логічно пов’язані рівні:

– на рівні веб-додатків знаходяться компоненти, які відповідають за відображення та взаємодію з користувачем; основною технологією цього рівня є бібліотека React, яка забезпечує динамічне оновлення контенту на сторінці без перезавантаження;

– на рівні даних, який відповідає за керування станом додатку та зберігання даних, використовується бібліотека Redux; ця бібліотека забезпечує централізоване управління станом та спрощену взаємодію з компонентами на рівні веб-додатків;

– на рівні інтеграції відбувається взаємодія з сервером через REST API, який дозволяє отримувати та відправляти дані між клієнтом та сервером за допомогою стандартних HTTP-запитів; це забезпечує ефективний обмін даними та високу швидкість роботи додатку.

Серверна частина застосунку ґрунтується на трьохрівневій архітектурі. Головна перевага полягає в можливості легко змінювати кожен рівень незалежно від інших, що робить систему більш гнучкою та легшою в обслуговуванні.

Ця архітектура складається з наступних рівнів:

– рівень уявлення;

– рівень бізнес-логіки;

– рівень доступу до даних.

Рівень представлення відповідає за те, як дані подаються користувачеві та як користувач взаємодіє з системою. На цьому рівні знаходяться контролери та шлюзи, що розроблені за допомогою фреймворка NestJS. У цьому рівні не міститься логіка бізнес-процесів або доступ до даних.

У NestJS контролери призначені для обробки HTTP-запитів та відповідей. Головна їх роль полягає в прийомі вхідних HTTP-запитів від клієнтів, їх обробці та

поверненні відповідей. Щодо шлюзів, вони використовуються для обробки та маршрутизації WebSocket-запитів.

Рівень бізнес-логіки відповідає за обробку даних і виконання бізнес-процесів програми. Тут розташовані сервіси, що виконують операції, пов’язані з обробкою та зміною даних відповідно до бізнес-правил. Цей рівень не має прямого доступу до даних, але отримує дані з рівня представлення та передає їх на рівень доступу до даних для збереження або вилучення.

У серверній частині ігрового програмного застосунку у жанрі multiplayer party games присутній рівень доступу до даних. Однак, оскільки дані кімнат зберігаються в пам’яті застосунку, цей рівень складається з набору сервісів та класів, які забезпечують доступ до них. Цей рівень не містить бізнес-логіки; він складається виключно з операцій, пов’язаних з доступом до даних.

Проєкт було започатковано з метою створення доступного та захоплюючого способу проведення часу з друзями, особливо в умовах дистанційного спілкування. Ідея полягала в об’єднанні популярних жанрів ігор у єдиний застосунок, який би задовольнив різні смаки та уподобання користувачів, забезпечуючи при цьому простий та інтуїтивно зрозумілий інтерфейс.

1.3 Межі

Етапи тестування включатимуть тільки системне тестування. Яке передбачатиме тестування системи в цілому, щоб переконатися, що вона відповідає вимогам та очікуванням користувачів. Типи тестування включатимуть: функціональне тестування, тестування інтерфейсу користувача, тестування конфігурації. Функціональне тестування буде проводитися для перевірки правильності роботи основних функцій застосунку, таких як створення ігрових сесій, управління ролями гравців, синхронізація дій та генерація відповідей штучним інтелектом. Тестування інтерфейсу користувача буде спрямоване на

перевірку зручності та інтуїтивності користування застосунком. Тестування конфігурації буде визначати сумісність застосунку з різними пристроями та веб-браузерами.

Перелік функцій та можливостей об’єкта тестування включатиме усі функціональності, зазначені в специфікації проєкту, такі як керування ігровими сесіями, гравцями, ігровою логікою та синхронізацією дій між учасниками гри. Крім того, об'єкт тестування повинен підтримувати адаптацію до різних розширень екрану та різних веб-браузерів.

Припущення, зроблені під час розробки документу, включають в себе стабільність серверної та клієнтської частин, наявність доступу до API компанії OpenAI для інтеграції штучного інтелекту та правильне функціонування веб-інтерфейсу в різних веб-браузерах.

Ризики тестування включають потенційні проблеми з синхронізацією дій між гравцями, збої під час отримання відповіді від API штучного інтелекту, а також труднощі з сумісністю з різними веб-браузерами та розмірами екранів пристроїв.

Обмеження тестування можуть включати обмежену кількість доступних ресурсів для проведення тестів у реальному часі, нестабільність інтернет-з'єднання, обмежену кількість тестових пристроїв та обмежену кількість часу на виконання тестування.

1.4 Ідентифікація проєкту

Таблиця нижче визначає наявність документації, яка була використана для розробки тест-плану (табл. 1.1):

Таблиця 1.1 – Ідентифікація проєкту (таблиця виконана самостійно)

Документ	Створено або доступно	Отримано або розглянуто	Автор або ресурс
Специфікація вимог програмного забезпечення	Так	Так	Бухало В. О., Двугрошев А. О., Мірошніков Є. В., Калініченко О. Ю.

Специфікація вимог до програмного забезпечення допоможе тестувальникам краще зрозуміти функціональні вимоги продукту.

2 ВИМОГИ ДО ТЕСТУВАННЯ

У наведеному нижче переліку визначено ті елементи – функціональні вимоги та нефункціональні вимоги – які були визначені як цілі для тестування.

Функціональні вимоги:

– перевірка можливості створення приватної ігрової кімнати після вибору однієї з міні-ігор зі слайдера;

– перевірка можливості підключення, відключення та перепідключення до ігрової сесії;

– перевірка можливості змінити статус ігрової сесії: спроба розпочати гру, поставити її на паузу та відновити після паузи;

– перевірка можливості зміни особистої інформації: нікнейму та аватару;

– перевірка коректності відображення даних, отриманих з сервера, конвертації часу та відображення етапів гри;

– перевірка розподілу функцій між різними користувачами відповідно до їх ролі;

– перевірка можливості зміни локалізації на контролері та хості, а також зміни локалізації на всіх підключених пристроях у приватній ігровій кімнаті;

– перевірка можливості виконання дій гравців у міні-грі Brain Knights: зміна команди, зміна назви команди, передача прав капітанства, вибір відповідального у раунді, вибір однієї з запропонованих відповідей, а також почергове виключення тем зі списку;

– перевірка можливості виконання дій гравців у міні-грі Turing Test: зміна команди, вписання та відправлення запитань, вписання та відправлення відповідей на запитання, голосування за певну відповідь зі списку;

– перевірка можливості виконання дій гравців у міні-грі Skibidy Party: завантаження власного пакету мемів, вибір та відправлення мему, розподіл призових місць суддею;

– перевірка зберігання та очищення даних ігрової сесії на сервері;

– перевірка налаштованості штучного інтелекту та його відповідей, що повинні залишитись передбачуваними, навіть у нестандартних ситуаціях.

Нефункціональні вимоги:

а) перевірка зручності використання:

– інтерфейс гри є інтуїтивно зрозумілим і легко освоюваним новими користувачами;

– всі важливі функції доступні з головного екрану або не більше ніж за 3 кліки;

б) перевірка сумісності:

– система коректно працює на всіх сучасних веб-браузерах (Chrome, Firefox, Edge, Safari, Yandex, Opera);

– програмне забезпечення пристосовується до різних розмірів екранів;

в) перевірка безпеки:

– система передає дані між клієнтом та сервером по захищеному HTTPS протоколу;

г) перевірка локалізації:

– система містить коректні переклади англійською та українською мовами.

Цей список представляє те, що буде протестовано.

3 СТРАТЕГІЯ ТЕСТУВАННЯ

3.1 Типи тестування

3.1.1 Функціональне тестування

Функціональне тестування має бути спрямоване на перевірку вимог, які можна безпосередньо пов'язати з варіантами використання, бізнес-функціями та бізнес-правилами. Метою таких тестів є впевнитися в правильному прийомі, обробці й пошуку даних, а також у коректному виконанні бізнес-правил. Це тестування використовує методи «чорної скриньки», тобто перевіряє програму та її внутрішні процеси шляхом взаємодії через графічний інтерфейс користувача (GUI) та аналізу результатів або вихідних даних.

Техніка тестування, його мета, критерії завершення та особливі аспекти функціонального тестування наведені в таблиці 3.1.

Таблиця 3.1 – Функціональне тестування (таблиця виконана самостійно)

Мета випробування	Забезпечити правильне функціонування об'єкта тестування, включаючи навігацію, введення, обробку та пошук даних
Критерії завершення	– всі заплановані тести були виконані; – усунуто всі виявлені дефекти;

Продовження таблиці 3.1

Особливі міркування	– перевірити, чи відповідає функціонал ігрового програмного застосунку у жанрі multiplayer party games вказаним функціональним вимогам; – переконатися, що всі бізнес-правила правильно виконуються; – забезпечити, щоб користувачі отримували відповідні повідомлення про помилки та інформацію при взаємодії з системою; – провести тестування кожного можливого сценарію використання, використовуючи як достовірні, так і недостовірні дані; це дозволить переконатися, що очікувані результати виникають при використанні достовірних даних, та забезпечить відображення відповідних повідомлень про помилки при використанні недостовірних даних.

Кінець таблиці 3.1

Техніка	Виконати кожен можливий сценарій використання, потік або функцію, використовуючи як достовірні, так і недостовірні дані, для перевірки наступного: – очікувані результати виникають при використанні достовірних даних; – відповідні повідомлення про помилки або попередження зёявляються у випадку використання некоректних даних; – кожне бізнес-правило застосовується коректно.

Ця таблиця сприяє організації процесу тестування та забезпечує систематичний підхід до перевірки функціональних можливостей системи. Ця систематизація полегшує тестерам оцінку відповідності продукту вимогам і виконання необхідних корекцій для гарантування його якості та надійності перед випуском на ринок.

3.1.2 Тестування інтерфейсу користувача

Перевірка інтерфейсу користувача (UI) є ключовим етапом у розробці програмної системи, що спрямований на оцінку правильності та зручності взаємодії користувача з програмним продуктом.

У таблиці 3.2 наведено методи, цілі випробування, критерії завершення та особливі відомості щодо тестування інтерфейсу користувача.

Таблиця 3.2 – Тестування інтерфейсу користувача (таблиця виконана самостійно)

Мета випробування	Виконати перевірку відповідності користувацького інтерфейсу системи вимогам, що забезпечить зручну навігацію та ефективну взаємодію користувача з функціоналом системи. Також це тестування спрямоване на переконанняся, що всі елементи і компоненти інтерфейсу працюють стабільно і відповідають встановленим стандартам якості, у тому числі корпоративним та галузевим стандартам
Техніка	Техніка випробування користувацького інтерфейсу спрямована на оцінку правильності та зручності взаємодії користувача з програмним забезпеченням. Один з важливих аспектів цієї техніки - це перевірка навігації в об'єкті випробування. Вона включає переходи між різними сторінками, а також взаємодію з окремими елементами інтерфейсу

Кінець таблиці 3.2

Критерії завершення:	– коректність переходів: всі переміщення між сторінками мають відбуватися правильно та без будь-яких затримок; – зручність взаємодії: користувач повинен з легкістю зрозуміти, як взаємодіяти з окремими елементами інтерфейсу та використовувати різні методи доступу, такі як клавіші, рух миші та натискання на екран смартфону; – відповідність вимогам: всі функції навігації мають відповідати вимогам, визначеним у специфікації, а також бізнес-функціям.
Особливі міркування:	– безпека: деякі можливості можуть бути обмежені для захисту від несанкціонованого доступу або зловмисних атак; – конфіденційність: доступ до певних даних може бути обмежений для збереження конфіденційності інформації користувачів; – стабільність: можуть бути введені обмеження для забезпечення стійкості та надійності програмного продукту.

Ця таблиця сприяє організації процесу тестування, забезпечуючи систематичний підхід до перевірки функціональних можливостей системи. Ця структуризація полегшує тестерам оцінку відповідності продукту вимогам та внесення необхідних змін для забезпечення його якості та надійності перед випуском на ринок.

3.1.3 Тестування конфігурації

Тестування конфігурацій перевіряє функціонування веб-додатка та серверної частини на різних наборах апаратних та програмних характеристик. Це включає тестування сумісності з різними версіями операційних систем, веб-браузерів, а також різними версіями та налаштуваннями серверного середовища.

У таблиці 3.3 наведені методи, цілі випробування, умови завершення та особливості, які стосуються тестування конфігурацій.

Таблиця 3.3 – Тестування конфігурації (таблиця виконана самостійно)

Мета випробування	Перевірити правильну роботу веб-додатка та серверної частини на різних апаратних та програмних конфігураціях
Критерії завершення	У всіх комбінаціях тестового та не-тестового програмного забезпечення всі функції успішно виконуються без виникнення помилок або некоректної поведінки

Кінець таблиці 3.3

Техніка	– тестування веб-додатку у різних веб-браузерах (Chrome, Firefox, Safari, Edge) на різних операційних системах (Windows, macOS, Linux); – запуск серверної частини на різних конфігураціях серверів з різними версіями Node.js та операційних систем.
Особливі міркування	– важливо враховувати різноманітність апаратних та програмних конфігурацій, включаючи версії операційних систем, типи пристроїв та розширення їх екранів; – необхідно встановити основний набір конфігурацій для тестування, який враховуватиме основні характеристики цільових аудиторій.

Ця таблиця спрощує організацію процесу тестування, надаючи рамки для систематичного аналізу функціональних можливостей системи. Цей структурований підхід дозволяє тестерам ефективніше оцінювати відповідність продукту вимогам та вносити необхідні зміни для гарантування його якості та надійності перед випуском на ринок.

3.2 Інструменти

Для забезпечення високоякісного тестування програмного забезпечення цього проєкту планується використання інструментів, наведених у таблиці 3.5.

Таблиця 3.5 – Інструменти (таблиця виконана самостійно)

	Інструмент	Постачальник/власна розробка	Версія
Управління тестуванням	Jira	Atlassian	9.13
Управління тестуванням	Google Sheets	Google LLC	–
Інструмент для тестування веб-API	Postman	Postman	11

Кожен інструмент з цього переліку дозволяє допомогти в проведенні тестування кожного компонента програмного забезпечення.

4 РЕСУРСИ

4.1 Ролі

Кадрові припущення для проєкту наведені в таблиці 4.1.

Таблиця 4.1 – Кадрові припущення для проєкту (таблиця виконана самостійно)

Працівники		
Посада	Рекомендовані мінімальні ресурси (кількість штатних ролей, призначених)	Конкретні обов’язки або коментарі
Тестувальник серверної частини проєкту	2	Виконати весь цикл тестування на сервері
Тестувальник клієнтської частини проєкту	2	Виконати весь цикл тестування на клієнті

Ця таблиця описує посаду, кількість та обов’язки працівників.

4.2 Система

У таблиці 4.2 наведено системні ресурси для проєкту тестування:

Таблиця 4.1 – Системні ресурси (таблиця виконана самостійно)

Системні ресурси	
Ресурс	Назва / Тип
Node.js	Node.js v20.10.0

Кінець таблиці 4.1

Системні ресурси	
Ресурс	Назва / Тип
React	React v18.2.0
Браузер	Google Chrome 112.0.5615.138, Mozilla Firefox 112.0, Microsoft Edge 123.0.2420.97, Safari 16.4, Opera 98.0.4759.15, Yandex 23.3.2
Операційна система	Windows 10, Windows 11, Android 13, iOS 17
Монітор	Full HD монітор, UltraWide WQHD монітор, Quad HD монітор
Персональний комп’ютер	Intel Core i5 9600k, 16 ГБ DDR4 RAM, NVIDIA GeForce RTX 2060 SUPER; Intel Core i5 12400, 32 ГБ DDR4 RAM, AMD Radeon 6800; Intel Core i5 13600kf, 32 ГБ DDR5 RAM, NVIDIA GeForce RTX 4070; AMD Ryzen 5 3600, 4 ГБ DDR4 RAM, GeForce GTX 1080
Смартфон	Xiaomi Redmi Note 11, Xiaomi Redmi Note 11 Pro 5G, iPhone XR, iPhone 12

Ця таблиця описує наступні ресурси: Node.js, React, браузер, операційна система, монітор, персональний комп’ютер та смартфон.

5 ЕТАПИ ПРОЄКТУ

Для забезпечення якості ігрового програмного застосунку у жанрі multiplayer party games, тестування буде проведено на різних етапах розробки. Етапи проєкту можна побачити в таблиці 5.1.

Таблиця 5.1 – Етапи проєкту (таблиця виконана самостійно)

Етапне завдання	Зусилля	Дата початку	Дата закінчення
Планування тесту	3	20.04.2024	26.04.2024
Проєктування тесту	4	26.04.2024	01.05.2024
Впровадження тесту	3	01.05.2024	06.05.2024
Виконання тесту	5	06.05.2024	16.05.2024
Оцінка тесту	3	16.05.2024	20.05.2024

Кожен етап тестування включатиме специфічні тестові заходи, які відповідають вимогам та функціональності проєкту, визначеним у попередніх розділах.

6 РЕЗУЛЬТАТИ

6.1 Тестова модель

Тест-план – це основний документ, який визначає стратегію тестування, область застосування, ресурси, ризики, графік та інші аспекти тестового процесу.

Звіт про виконання тестів – цей звіт містить інформацію про кількість виконаних тестів, кількість успішно пройдених тестів, кількість виявлених дефектів та їх статус.

6.2 Журнали випробувань

Для запису та звітування про результати тестування і статус тестування буде використовуватися система журналів випробувань, яка включатиме такі компоненти:

а) інструменти:

1) система журналів випробувань буде реалізована з використанням інтегрованих функцій у розробницькому середовищі Visual Studio Code;

б) методи:

1) журнали випробувань будуть вести всі члени команди, які беруть участь у тестуванні.

6.3 Звіти про дефекти

Для відстеження дефектів та їх статусу буде використана система керування задачами, така як Jira. Крім того, для відстежування багів буде використовуватися

Google Excel. Кожен виявлений дефект буде документований у системі керування задачами, включаючи детальний опис проблеми, кроки для відтворення, пріоритет та відповідального за виправлення. Дефекти будуть регулярно оглядатися командою розробників та тестувальників для визначення статусу та пріоритету виправлення.

ДОДАТОК А

Завдання проєкту

Нижче наведені завдання, пов’язані з тестом:

а) спланувати тест:

1) визначити вимоги до тесту;
2) оцінити ризики;
3) розробити стратегію тестування;
4) визначити тестові ресурси;
5) створити розклад;
6) сформувати тест-план;

б) спроєктувати тест:

1) підготувати аналіз робочого навантаження;
2) визначити та описати тест-кейси;
3) визначити та структурувати тестові процедури;
4) проаналізувати та оцінити тестове покриття;

в) впровадити тест:

1) записати або запрограмувати тестові скрипти;
2) визначити специфічну для тестів функціональність у моделі проєктування та реалізації;
3) створити зовнішні набори даних;

г) виконати тест:

1) виконати тестові процедури;
2) оцінити виконання тесту;
3) відновити тест після його зупинки;
4) перевірити результати;
5) дослідити неочікувані результати;
6) зареєструвати дефекти;

д) оцінити тест:

1) оцінити покриття тест-кейсів;

2) оцінити покриття коду;

3) проаналізувати дефекти;

4) визначити, чи були досягнуті критерії завершення тесту та критерії успіху.

ДОДАТОК Д

Тест-кейси клієнтської частини ігрового програмного застосунку у жанрі multiplayer party games

Таблиця Д.1 – Тест-кейс №2 (таблиця виконана самостійно)

Інформація про тест-кейс			
Ідентифікатор тесту:	Тест-кейс №1		
Опис функції:	Перевірка функції завантаження свого набору мемів в грі «Skibidy Party»		
Власник тесту:	Двугрошев Андрій Олексійович		
Дата створення:	20.05.2024		
Мета тесту:	Перевірити коректність роботи функції завантаження набору файлів в грі «Skibidy Party»		
Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	З використанням клієнтської частини створено ігрову сесію гри «Skibidy Party»;	Ігрову сесію створено, і її дані збережено в пам’яті серверного застосунку;	Пройдено
2	З використанням клієнтської частини до створеної кімнати підключено чотирьох гравців;	Серверний застосунок записав в лог подію підключення кожного гравця, а дані гравців додаються до масиву та зберігаються в пам'яті застосунку;	Пройдено
3	З використанням клієнтської частини перейти до меню завантаження набору користувацьких мемів	Клієнтська частина змінює інтерфейс на меню завантаження мемів;	Пройдено

Кінець таблиці Д.1

Передумова			
№	Опис випадку	Очікуваний результат	Висновок
4	З використанням клієнтської частини вибрати меми з меню файлової системи комп'ютера;	Серверний застосунок записав в лог подію завантаження мемів та зберіг до пам’яті меми які будуть використовуватись в грі. Сервер створює нову сесію;	Пройдено
Результати тестування			
Тестувальник: Двугрошев А. О.	Дата прогону тесту: 20.05.2024	Результат тесту (P/F/B): ПРОЙДЕНО (P)	

Таблиця Д.2 – Тест-кейс №2 (таблиця виконана самостійно)

Інформація про тест-кейс	
Ідентифікатор тесту:	Тест-кейс №2
Опис функції:	Перевірка перенесення ролі «Party owner» при виході гравця з нею на іншого користувача у ігровій сесії
Власник тесту:	Двугрошев Андрій Олексійович
Дата створення:	19.05.2024
Мета тесту:	Перевірити коректність обробки запиту на зміну статусу гравця після виходу гравця з роллю «Party owner»

Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	З використанням клієнтської частини створено ігрову сесію будь-якої гри застосунку	Ігрову сесію створено, і її дані збережено в пам’яті серверного застосунку;	Пройдено

Кінець таблиці Д.2

Передумова			
№	Опис випадку	Очікуваний результат	Висновок
2	З використанням клієнтської частини до створеної кімнати підключено мінімальну кількість гравців;	Серверний застосунок записав в лог подію підключення кожного гравця, а дані гравців додаються до масиву та зберігаються в пам'яті застосунку;	Пройдено
3	З використанням клієнтської частини підтверджено готовність усіх гравців до гри;	Серверний застосунок записав в лог подію зміни готовності до гри кожного гравця, а дані готовності гравців були змінені та збережені в пам’яті застосунку;	Пройдено
4	Власник кімнати розпочав гру, використовуючи клієнтську частину;	Серверний застосунок записав в лог подію початку гри;	Пройдено
5	Власник кімнати виходить з ігрової сесії використовуючи мобільний ігровий контролер	Серверний застосунок записав в лог подію зміни статусу гравця на відключений, та робить запит на зміну ролі другого за часом входу до сесії з минулої ролі на «Party owner»	Пройдено

Результати тестування		
Тестувальник: Двугрошев А. О.	Дата прогону тесту: 20.05.2024	Результат тесту (P/F/B): ПРОЙДЕНО (P)

Таблиця Д.3 – Тест-кейс №3 (таблиця виконана самостійно)

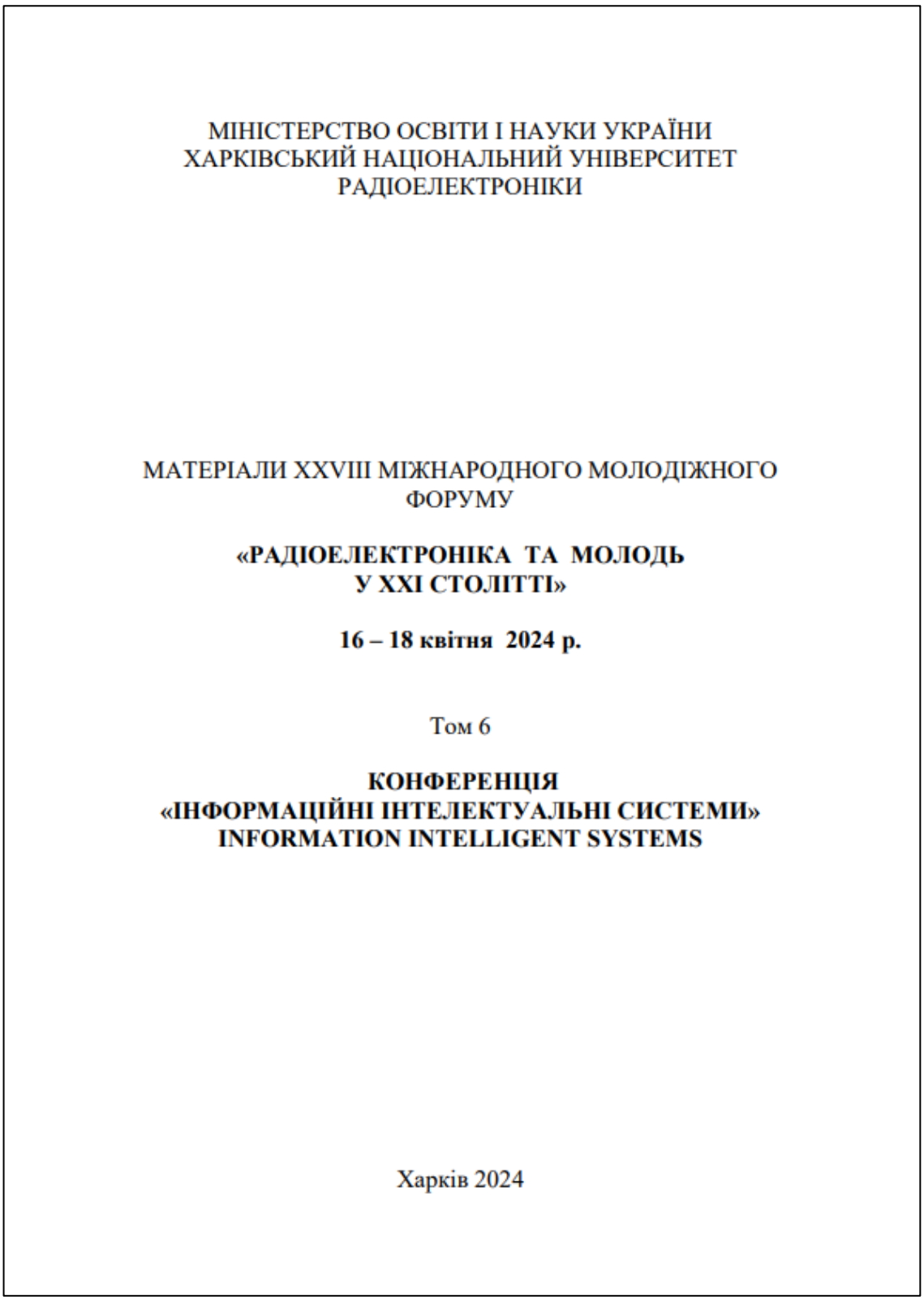
Інформація про тест-кейс			
Ідентифікатор тесту:	Тест-кейс №3		
Опис функції:	Перевірити функцію паузи на клієнтській частині гри		
Власник тесту:	Двугрошев Андрій Олексійович		
Дата створення:	20.05.2024		
Мета тесту:	Перевірити коректність роботи функції паузи на клієнтскій частині гри		
Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	З використанням клієнтської частини створено ігрову сесію будь-якої гри застосунку	Ігрову сесію створено, і її дані збережено в пам’яті серверного застосунку;	Пройдено
2	З використанням клієнтської частини до створеної кімнати підключено мінімальну кількість гравців;	Серверний застосунок записав в лог подію підключення кожного гравця, а дані гравців додаються до масиву та зберігаються в пам'яті застосунку;	Пройдено
3	З використанням клієнтської частини підтверджено готовність усіх гравців до гри;	Серверний застосунок записав в лог подію зміни готовності до гри кожного гравця, а дані готовності гравців були змінені та збережені в пам’яті застосунку;	Пройдено

Кінець таблиці Д.3

Передумова			
№	Опис випадку	Очікуваний результат	Висновок
4	З використанням клієнтської частини підтверджено готовність усіх гравців до гри;	Серверний застосунок записав в лог подію зміни готовності до гри кожного гравця, а дані готовності гравців були змінені та збережені в пам’яті застосунку;	Пройдено
5	Власник кімнати розпочав гру, використовуючи клієнтську частину;	Серверний застосунок записав в лог подію початку гри;	Пройдено
3	На клієнтскій частині гри користувач активує функцію паузи	Серверний застосунок відправляє запит на зупинку таймеру та всіх ігрових механік	Пройдено
4	На клієнтській частині грі користувач активує функцію поновлення гри	Серверний застосунок відправляє запит на відновлення таймеру та всіх ігрових механік	Пройдено
Результати тестування			
Тестувальник: Двугрошев А. О.	Дата прогону тесту: 20.05.2024	Результат тесту (P/F/B): ПРОЙДЕНО (P)	

ДОДАТОК Е

Конференція «Інформаційні інтелектуальні системи» на 28-му Міжнародному форумі «Радіоелектроніка та молодь у XXI столітті»



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
РАДІОЕЛЕКТРОНІКИ

МАТЕРІАЛИ XXVIII МІЖНАРОДНОГО МОЛОДІЖНОГО ФОРУМУ

«РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ У XXI СТОЛІТТІ»

16 – 18 квітня 2024 р.

Том 6

КОНФЕРЕНЦІЯ
«ІНФОРМАЦІЙНІ ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ»
INFORMATION INTELLIGENT SYSTEMS

Харків 2024

Рисунок Е.1 – Титульна сторінка (знімок екрана виконано самостійно)

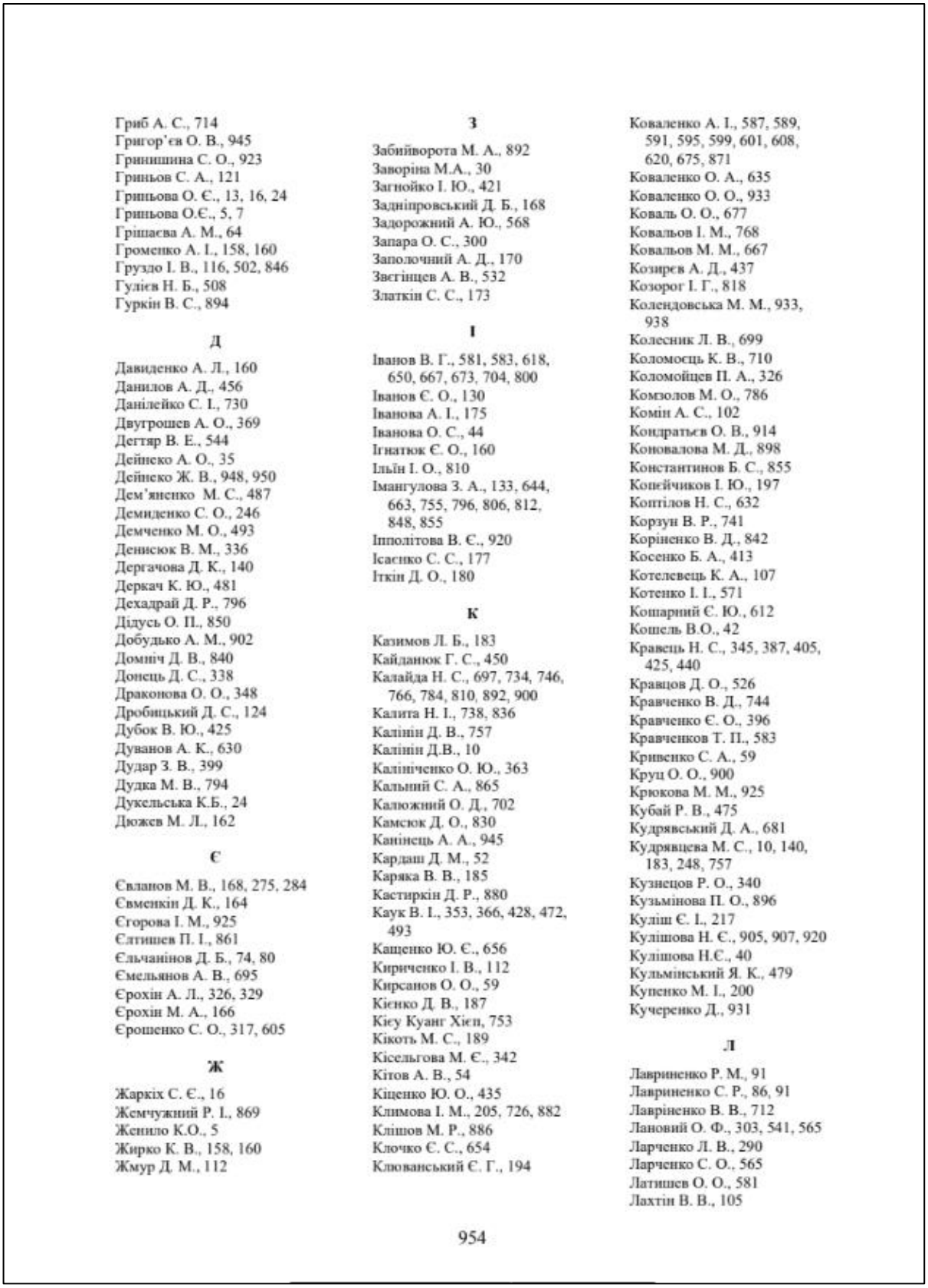


Рисунок Е.2 – Алфавітний покажчик (знімок екрана виконано самостійно)

УДК 004.514 DOI: https://doi.org/10.30837/IYF.IIS.2024.369

ВИКОРИСТАННЯ СМАРТФОНА З ВЕБ-ЗАСТОСУНКОМ ЯК БЕЗДРОТОВОГО КОНТРОЛЕРА В БАГАТОКОРИСТУВАЦЬКИХ ІГРАХ НА ОСНОВІ ТЕХНОЛОГІЇ ПОСТІЙНОГО ПІДКЛЮЧЕННЯ WEBSOCKET

Двугрошев А. О.
Науковий керівник – ст. викл. Новіков Ю. С.
Харківський національний університет радіоелектроніки, каф. ПІ,
м. Харків, Україна
e-mail: andrii.dvuhroshev@nure.ua

The current work explores the use of a mobile web application as a game controller using WebSocket technology. The development of such a tool involves the transfer of the main game logic to the server, while the environment display client computes the entire visualization of the game, significantly reducing the load on the overall system. Using mobile game controller systems, in turn-based games the game mechanic ensures efficient execution of in-game events with minimal feedback from the system. The work used the Node.js runtime environment, the React library, and the TypeScript and JavaScript programming languages.

За останні десятиліття мобільні пристрої стали неодмінною частиною нашого повсякденного життя, а їх потужності та можливості продовжують зростати. У той же час відбувається стрімкий розвиток веб-технологій, що відкриває нові можливості для створення інноваційних додатків та сервісів. Впровадження мобільного пристрою до ігрового процесу відкриває перед розробниками можливості створення багатокористувацького ігрового застосунку з унікальними та передовими механіками.

Дана робота спрямована на реалізацію програмного коду, призначеного для використання в межах ігрового додатку, що забезпечує ефективне керування ігровим процесом за допомогою смартфону з мобільним застосунком, та виведення ігрового оточення у єдиному екрані браузеру. Для реалізації програмного модулю використовується технологія WebSocket [1] як протокол постійного підключення сервера до клієнта.

Веб-застосунок для смартфону реалізований за допомогою бібліотеки React для розробки інтерфейсу контролера, та методів React Hooks, оскільки вони є більш оптимізованими з точки зору продуктивності, ніж альтернативний підхід з використанням методів Redux [2], що важливо для мінімізації затримки відгуку від смартфона.

Перевагою такого методу управління стане легкодоступність та простота використання. Така реалізація контролера, добре підходить для багатокористувацьких ігор з виведенням ігрового процесу на один екран, в яких час відгуку не впливає на якість ігрового досвіду. Прикладом

Рисунок Е.3 – Перша сторінка тез (знімок екрана виконано самостійно)

реалізації управління через смартфон може стати контролер для гри в багатокористувацьку вікторину, в якій користувачі можуть вибирати правильну відповідь через смартфон, гоночна аркада – користувач використовує гіроскоп для керування транспортом, просторові головоломки – гравець використовує сенсорне управління і гіроскоп для вирішення головоломки змагаючись з іншими на швидкість.

Однак метод управління смартфоном, може бути незручний у багатокористувацьких іграх з важливістю малого відгуку від контролера – екшн ігри, стратегії, аркадні файтинги. Оскільки при такому методі затримка відчутно більша, якщо порівнювати з класичними методами управління ігровим процесом.

Програмний модуль забезпечує наступні можливості:

– налаштування відображення ігрового оточення та можливість контролю стану гри за допомогою пристроя з браузером що підтримує JavaScript;

– можливість підключення великій кількості гравців;

– зміну виконуваної події у реальному часі;

– можливість перепідключення смартфону;

– розподілення ігрового навантаження рівномірно на три модулі: сервер, клієнт, екран ігрового оточення.

Це досягається за допомогою розподілення обчислювальної логіки на 3 окремих підмодулі: додаток на смартфоні збирає вихідну інформацію та передає її на сервер, де в свою чергу підраховується основна ігрова логіка, після обробки він передає дані до клієнтської частини гри де відображається ігрове оточення, браузерний клієнт в такій системі бере на себе навантаження щодо прорахунку всього графічного оточення.

Принцип роботи обміну даними графічно наведено на рисунку 1.

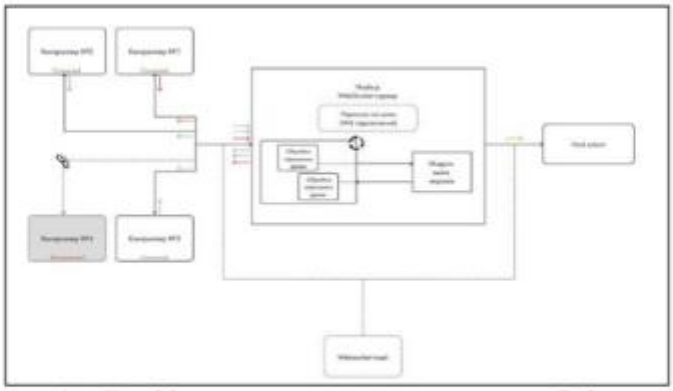


Рисунок 1 – Графічне представлення системи обміну даними

Затримка відгуку в такій системі варіюватиметься від 5 до 40 мілісекунд залежно від навантаження на сервер. Ця система добре

Рисунок Е.4 – Друга сторінка тез (знімок екрана виконано самостійно)

підходить в іграх із малозначущою ігровою затримкою – вікторини, головоломки, паті ігри, або ігри з системою split-screen. Модульність дає можливість легко масштабувати систему і змінювати її при потребі.

На основі розробки програмного коду можна зробити висновок, що запропонована система ефективно використовує смартфон з мобільним веб-застосунком як контролер гри, використовуючи технологію WebSocket для забезпечення постійного підключення сервера до клієнта. Розподіл обчислювальної логіки на серверній, клієнтській частині та мобільного додатку, сприяє оптимізації навантаження та забезпечує швидку відповідь системи на дії гравців. Завдяки смартфонам, обладнаним мобільним веб-застосунком та використовуючи технологію WebSocket для забезпечення постійного підключення до сервера, гравці отримують можливість контролювати різноманітні аспекти гри. Наприклад, вони можуть обирати відповіді у багатокористувацьких вікторинах, керувати станом гри (налаштування, пауза, пуск гри), а також використовувати сенсорне управління у іграх-головоломках. Цей підхід дозволяє максимально використовувати потенціал сучасних мобільних пристроїв у ігрових додатках та відкриває широкі перспективи для подальшого розвитку і вдосконалення ігрового досвіду на основі веб-технологій. Використані ігрові механіки також демонструють високу ефективність в умовах значної затримки відгуку системи, та не впливають на досвід гри. Завдяки модульності система може легко масштабуватись та адаптуватись до різних типів ігор та вимог користувачів. Загальні результати дослідження свідчать про успішну реалізацію поставленої мети створення ефективного мобільного контролера для ігор на базі веб-технологій. Дана робота відкриває перспективи для подальших досліджень у цьому напрямку та можливості використання розробленої системи в різних ігрових середовищах.

Список використаних джерел:

1. Melnikov A. RFC 6455: The WebSocket Protocol. IETF Datatracker. URL: https://datatracker.ietf.org/doc/html/rfc6455 (дата звернення: 24.03.2024).

2. Pronina D., Kyrychenko I. Comparison of Redux and React Hooks Methods in Terms of Performance / D. Pronina, I. Kyrychenko // Proceedings of the 6th International Conference on Computational Linguistics and Intelligent Systems (COLINS 2022). Volume I: Main Conference, 12–13 May 2022. – Gliwice : CEUR Workshop Proceedings, 2022. – № 3171. – P. 791–800.

Рисунок Е.5 – Третя сторінка тез (знімок екрана виконано самостійно)

ДОДАТОК Ж

Отримана нагорода на виставці технічної творчості молоді на 28-му Міжнародному форумі «Радіоелектроніка та молодь у XXI столітті»


МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

ДИПЛОМ

НАГОРОДЖУЄТЬСЯ

Калініченко Олександр, Двугрошев Андрій,
Бухало Володимир, Мірошніков Єгор

за II місце

в номінації «Ігрові технології»
на виставці технічної творчості молоді
28-го Міжнародного молодіжного форуму
«Радіоелектроніка та молодь в XXI столітті»

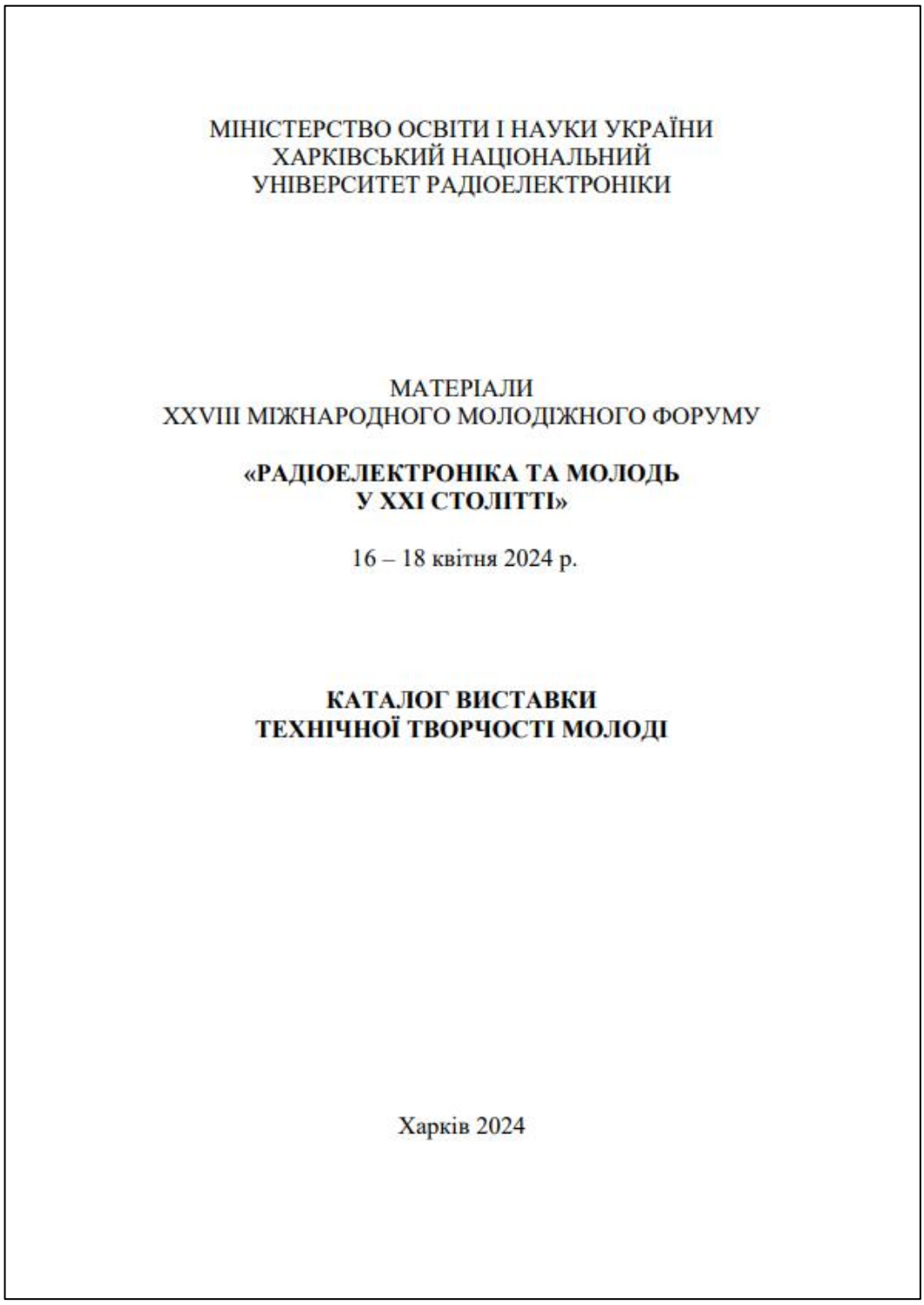
Голова оргкомітету Форуму Ігор РУБАН

18 квітня 2024 р.

Рисунок Ж.1 – Отриманий диплом (знімок екрана виконано самостійно)

ДОДАТОК И

Виставка технічної творчості молоді на 28-му Міжнародному форумі «Радіоелектроніка та молодь у XXI столітті»



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

МАТЕРІАЛИ
XXVIII МІЖНАРОДНОГО МОЛОДІЖНОГО ФОРУМУ

**«РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ
У XXI СТОЛІТТІ»**

16 – 18 квітня 2024 р.

**КАТАЛОГ ВИСТАВКИ
ТЕХНІЧНОЇ ТВОРЧОСТІ МОЛОДІ**

Харків 2024

Рисунок И.1 – Титульна сторінка (знімок екрана виконано самостійно)

4. Комп'ютерна гра «Close Space

Автори: ***Масалов Максим Володимирович, Луппа Артем Дмитрович,*** ст. гр. КІУКІу-23-2, ХНУРЕ.

Науковий керівник: Малькова Ірина Анатоліївна, ас. каф. ІУС, ХНУРЕ.

Ігровий додаток призначений для розваги та корисного проведення вільного часу, дозволяє гравцям розвивати свої інтелектуальні здібності при виконанні різних видів завдань.

Гра виконана в жанрі асиметричного командного хоррора - жанр ігор, в яких гравці розділені на дві різні команди з різними цілями та можливостями.

Додаток розроблено за допомогою ігрового двигуна Unity, мови програмування C# та великої кількості бібліотек. З'єднання між гравцями забезпечено завдяки бібліотеці Photon.

Переваги розробки: кросплатформеність, легкість використання, оригінальний дизайн.

5. Гра «Шахи 3D»

Автор: ***Золотухін Микола Владиславович,*** ст. гр. КІУКІу23-2, ХНУРЕ.

Науковий керівник: Павленко Євген Петрович, к.т.н., доц. каф. АПОТ, ХНУРЕ.

Гра «Шахи 3D» - інтерактивний додаток призначений для інтелектуального розвитку людини, проведення вільного часу та відточування навичок у грі разом з друзями. Гра виконана у жанрі покрокової стратегії.

Додаток розроблено з використанням середовища Unity та мови програмування C#.

Переваги розробки: простий та зрозумілий інтерфейс, легкість у використанні.

6. Ігровий програмний застосунок в жанрі Multiplayer Party Games «ROFLSFUN»

Автори: ***Калініченко Олександр Юрійович, Двугрошев Андрій Олексійович, Бухало Володимир Олександрович, Мірошніков Єгор Вячеславович,*** ст. гр. ПЗПІ-20-10, ХНУРЕ.

Науковий керівник: Новіков Юрій Сергійович, старший викладач. каф. ПІ, ХНУРЕ.

Розроблено мультиплеєрний ігровий додаток з використанням фреймворку React для клієнтської частини та програмної платформи Node.js для серверної частини. Для забезпечення зв'язку між клієнтом та сервером використовується технологія WebSocket.

Гра включає в себе три міні-ігри: «Skibidy Party», «Brain Knights» та «Turing Test». Додаток запускається з сайту «roflsfun.onrender.com», створюючи приватну ігрову кімнату з випадковим чотирьохзначним ключем. Гравці можуть приєднатися до неї через мобільні пристрої, вводячи ключ кімнати та свій нікнейм.

У грі є мінімальна та максимальна кількість гравців, перший, хто приєднується до кімнати, стає її лідером і може розпочати гру, коли набрана мінімальна кількість гравців. Гра складається з N раундів, де кожен має можливість набрати певну кількість балів. Переможцем становиться той, хто набрав найбільшу кількість балів. У кінці гри, кожен гравець отримує певний титул за досягнення.

15

Рисунок И.2 – Каталог виставки (знімок екрана виконано самостійно)

ДОДАТОК К

Отримана нагорода на конференції «Інформаційні інтелектуальні системи» на 28-му Міжнародному форумі «Радіоелектроніка та молодь у XXI столітті»


ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

ГРАМОТА

НАГОРОДЖУЄТЬСЯ

Двугрошев Андрій Олексійович

(група ПЗПІ-20-10,
Харківський національний університет радіоелектроніки),

за змістовну доповідь

«Використання смартфона з веб-застосунком
як бездротового контролера в багатокористувацьких іграх
на основі технології постійного підключення WEBSOCKET»

на секції

«Програмна Інженерія. Інформаційні технології в освіті»

конференції

«ІНФОРМАЦІЙНІ ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ»

в рамках **28-го** Міжнародного молодіжного форуму

«РАДІОЕЛЕКТРОНІКА І МОЛОДЬ В XXI СТОЛІТТІ»

Голова конференції
«Інформаційні інтелектуальні
системи», д.т.н., проф. Андрій ЄРОХІН

м. Харків, 2024

Рисунок К.1 – Отримана грамота (знімок екрана виконано самостійно)