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 у жанрі Multiplayer Party Games. Контролер, баланс ігрового процесу
(тема)

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

Мірошніков Є. В.
(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 Мірошнікову Єгору Вячеславовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Ігровий програмний застосунок в жанрі Multiplayer Party Games. Контролер, баланс ігрового процесу
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) _____ Мірошніков Є. В.
(підпис)

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

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, 166 стор., 40 рис., 9 додатків, 1 табл., 12 джерел.

БАГАТОКОРИСТУВАЦЬКІ ІГРИ ДЛЯ ВЕЧІРОК, БАЛАНСУВАННЯ ІГРОВОГО ПРОЦЕСУ, ІГРОВИЙ ПРОГРАМНИЙ ЗАСТОСУНОК, КЛІЄНТСЬКА ЧАСТИНА, JAVASCRIPT, NODE.JS, REACT, SOCKET.IO, TYPESCRIPT

Об’єкт розробки – мобільний програмний ігровий контролер, балансування ігрового процесу в ігровому програмному застосунку в жанрі multiplayer party games.

Мета розробки – реалізувати мобільний ігровий контролер для управління ігровим процесом та розробити баланс ігрового процесу в ігровий програмний застосунок, який буде містити три міні-гри в жанрі multiplayer party games.

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

У результаті розробки реалізований баланс відповідно часу та складності, мобільний ігровий контролер у клієнтській частині системи, який містить три міні-гри в жанрі multiplayer party games.

MULTIPLAYER PARTY GAMES, GAMEPLAY BALANCING, GAME SOFTWARE APPLICATION, CLIENT SIDE, JAVASCRIPT, NODE.JS, REACT, SOCKET.IO, TYPESCRIPT

The object of development is a mobile software game controller, balancing the gameplay in a game software application in the genre of multiplayer party games.

The goal of the development is to implement a mobile game controller to control the gameplay and develop a balance of gameplay in a gaming software application that will contain three mini-games in the genre of multiplayer party games.

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

The development resulted in a balance of time and complexity, a mobile game controller in the client part of the system, which contains three mini-games in the genre of multiplayer party games.

Я, Мірошніков Єгор Вячеславович, студент гр. ПЗПІ-20-10, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Ігровий програмний застосунок у жанрі Multiplayer Party Games. Контролер, баланс ігрового процесу», 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

UML – Unified Modeling Language

ПЗ – Програмне Забезпечення

SPA – Single Page Application

API – Application Programming Interface

UI – User Interface

UX – User Experience

HTTP – HyperText Transfer Protocol

DOM – Document Object Model

QR – Quick Response

TLS – Transport Layer Security

CDN – Content Delivery Network

UUID – Universally Unique Identifier

URL – Uniform Resource Locator

ВСТУП

Розвиток ігрової індустрії в останні десятиліття зазнав значного прогресу, пропонуючи гравцям широкий спектр ігрових жанрів та інтерактивних досвідів. Одним з найпопулярніших жанрів є multiplayer party games, які пропонують веселі та захоплюючі ігри для спільного проведення часу з друзями та родиною. Однак, щоб забезпечити тривалий інтерес гравців, необхідно не лише цікавий геймплей, а й його ефективне балансування, привабливий дизайн та інше.

Тема кваліфікаційної роботи спрямована на аналіз та оптимізацію ігрового досвіду, а також у створенні ефективної системи, що дасть можливість користувачу комфортно взаємодіяти з ігровим програмним застосунком та створення ігрового оточення що дасть можливість швидко орієнтуватись у грі та зручно управляти ігровими сценами у multiplayer party games. Основний акцент роботи зроблено на балансуванні складності ігор та розробку мобільного ігрового контролера з метою створення захоплюючого та справедливого ігрового середовища.

Метою роботи є проведення аналізу складності ігор, визначення ключових аспектів для балансування геймплею та їх оптимальні стратегії. Також метою є розробка мобільного ігрового контролера, який буде реалізований на клієнтській частині системи. Цей компонент відповідатиме за управління ігровим процесом, відображення статистики гравця, навігації по екрану де буде відображений геймплей гри. Для досягнення цих цілей використовуються такі технології, як середовище розробки Visual Studio Code, бібліотеки React, програмна платформа Node.js та Socket.IO, а також мови програмування JavaScript на клієнтській частині проєкту та TypeScript на серверній частині.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Багатокористувацькі ігри для вечірок стрімко зростають у популярності, пропонуючи динамічне поєднання командної роботи, змагання та чистої забави для зустрічей з друзями та знайомими. Завдяки простим правилам і механікам, вони залучають гравців усіх рівнів, сприяючи створенню дружньої атмосфери як у командних, так і в змагальних умовах. Розвиток цього жанру віддзеркалює зростання соціальної активності в цифровому просторі, об’єднуючи гравців заради спільного досвіду та товариських стосунків.

Балансування ігрового процесу має важливе значення для створення справедливого та приємного ігрового досвіду для всіх гравців. Воно включає в себе налаштування різних ігрових елементів, таких як рівні складності, умови перемоги та здібності гравців, щоб забезпечити збалансовану конкуренцію та задоволення гравців [1]. Використовуючи різні інструменти, аспект балансування ігрового процесу проекту зосереджується на точному налаштуванні ігрової механіки та динаміки. Це включає налаштування таймерів та рівня складності для досягнення оптимального балансу.

Функціональність контролера в багатокористувацьких іграх для вечірок має вирішальне значення для забезпечення безперебійної та захопливої гри. Це передбачає підключення до ігрових сесій за допомогою унікальних кімнатних ключів, що дозволяє гравцям безперешкодно приєднуватися до ігор. Контролер відображає поточний стан гри та забезпечує взаємодію через веб-браузери на смартфонах чи інших пристроях. Він обробляє дані, отримані від сервера, щоб інформувати гравців про хід гри та необхідні дії. Забезпечуючи взаємодію та оновлення в режимі реального часу, контролер покращує загальний ігровий досвід та утримує гравців у грі.

Розробка multiplayer party games має свої переваги. Ці ігри сприяють соціальній взаємодії, розвиваючи товариські стосунки та обмін досвідом між

гравцями. Вони орієнтовані на широку аудиторію, в них можуть грати гравці всіх рівнів та віку. Завдяки простим правилам і механікам, вони доступні для новачків і водночас пропонують глибину для досвідчених геймерів. Крім того, багатокористувацькі ігри для вечірок ставлять на перше місце веселощі та розваги, забезпечуючи гравцям години насолоди. Багато з цих ігор наголошують на командній роботі та змаганні, заохочуючи співпрацю та дружнє суперництво між гравцями. Більше того, розробка таких ігор може призвести до створення яскравих спільнот, де гравці спілкуються, обмінюються стратегіями та формують міцні дружні стосунки. На ігровому ринку зростає попит, що дає розробникам можливість вийти на прибутковий ринок, який постійно розширюється. Нарешті, цей жанр дозволяє розробникам досліджувати інноваційні ігрові механіки, креативні наративи та унікальні художні стилі, розширюючи межі творчості та надаючи свіжий ігровий досвід.

Загалом, розробка multiplayer party games – це чудова можливість створити захопливий, розважальний та соціально значущий ігровий досвід.

1.2 Аналіз конкурентів

Порівнюючи ігровий програмний застосунок з такими конкурентами, як Jackbox Party та Gartic Phone, слід врахувати кілька факторів. Як Jackbox Party, так і Gartic Phone є визнаними гравцями на ринку багатокористувацьких ігор для вечірок, кожен з яких має свої сильні та слабкі сторони.

Jackbox Party – це серія комп’ютерних ігор, відома своєю різноманітністю та інтерактивністю. Вона пропонує широкий спектр розваг для гравців будь-якого віку та ігрового досвіду. Головна особливість цієї серії – це можливість грати з великою групою друзів або родичів, використовуючи лише свої смартфони або комп’ютери [2]. В ній існує безліч міні-ігор, кожна з яких має свої унікальні правила та механіки (див. рис. 1.1-1.2). Наприклад, деякі ігри – це вікторини, де

гравці відповідають на запитання на різні теми, збираючи бали за правильні відповіді. Інші ігри можуть включати змагання з малювання, де гравці намагаються передати певну ідею через малюнок, а потім інші гравці вгадують, що це за ідея.

Хоча Jackbox Party надає велику кількість розваг та можливість грати з великою групою друзів, деякі гравці можуть виявити, що гра не надає досить гнучкості у доступності або можливості налаштування правил для певних ігор.

Рисунок 1.1 – The Jackbox Party Pack (знімок екрана виконано самостійно)

Рисунок 1.2 – The Jackbox Party Pack 9 (знімок екрана виконано самостійно)

Gartic Phone – це онлайн-гра для малювання, яка здобула популярність завдяки своїй простоті та доступності. Головна ідея гри полягає в тому, щоб гравці чергово малювали задані слова або фрази, а потім інші гравці вгадували, що це за малюнок. Однією з ключових особливостей Gartic Phone є його простота. Інтерфейс гри і механіка малювання дуже інтуїтивні, що робить їх легко зрозумілими для гравців будь-якого віку (див. рис. 1.3-1.4). Величезний плюс полягає в тому, що гра не потребує встановлення жодного спеціального програмного забезпечення – вона запускається просто через веб-браузер. У Gartic Phone може брати участь велика кількість гравців одночасно, що робить її ідеальним варіантом для вечірок або онлайн-спілкування з друзями.

Однак, деякі гравці можуть виявити обмеженість гри через її простоту. Відсутність глибини в ігровому процесі та відсутність різноманітних режимів гри можуть спровокувати бажання досліджувати інші альтернативи для отримання більшої різноманітності в геймплей.

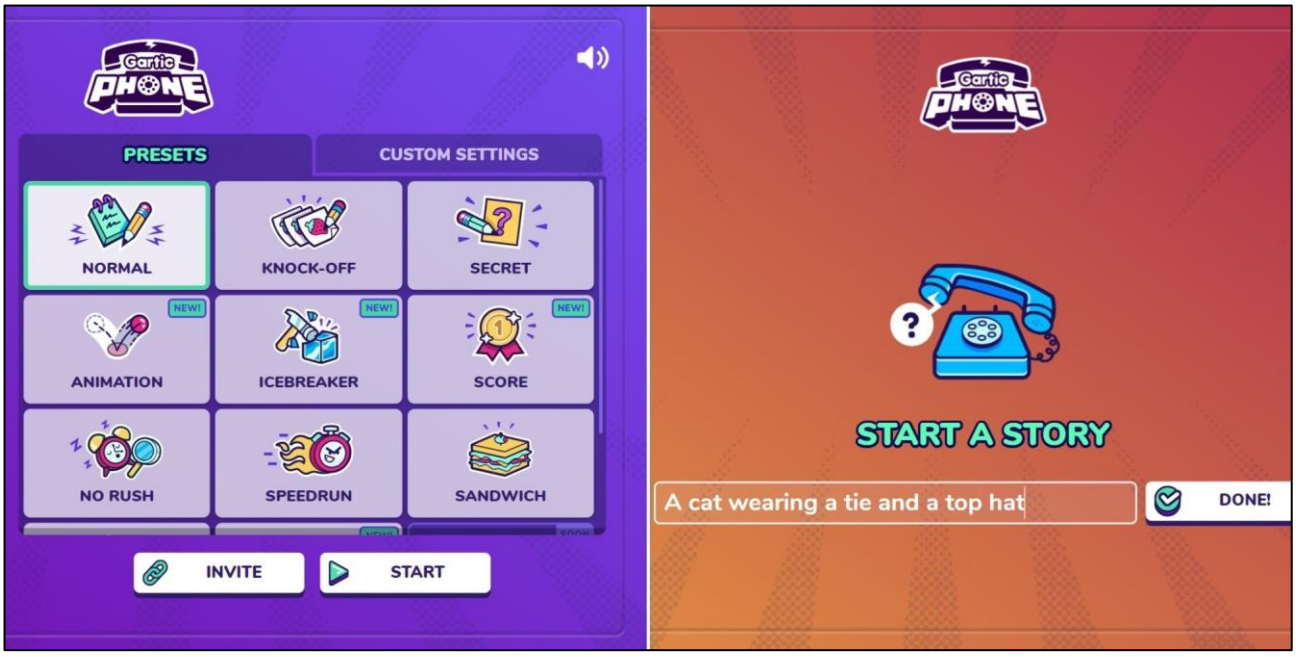


Рисунок 1.3 – Вибір режимів гри Gartic Phone (знімок екрана виконано самостійно)

Рисунок 1.4 – Ітерація гри Gartic Phone (знімок екрана виконано самостійно)

Чому наш продукт виділяється серед своїх конкурентів? По-перше, ігровий застосунок може похвалитися безпрецедентною доступністю, оскільки в неї можна грати на будь-якому пристрої з веб-браузером і підключенням до Інтернету. Це гарантує, що гравці можуть приєднатися до розваги незалежно від обраної платформи або пристрою. Крім того, пропонує різноманітну колекцію міні-ігор, кожна з яких пропонує унікальний і захоплюючий досвід. Від повсякденних завдань до середньовічних викликів і науково-фантастичних пригод, ігровий програмний застосунок в жанрі multiplayer party games пропонує широкий спектр вражень, щоб задовольнити різні уподобання гравців. Також, наш збірник поєднує в собі як командні, так і індивідуальні міні-ігри, що дозволяє проводити динамічні та змагальні ігрові сесії. Включення унікальних титулів в якості винагороди додає додатковий рівень стимулу для гравців прагнути до успіху. Продукт орієнтований на співпрацю, конкуренцію та навички командної роботи, що надає гравцям можливості для стратегічного мислення та взаємодії.

1.3 Виявлення та вирішення проблем

Майбутнє multiplayer party games багатообіцяюче, а високоякісні, оригінальні ігри будуть користуватися значним попитом. Розумні стратегії розвитку, що враховують сучасні тенденції та вподобання гравців, є запорукою стабільного успіху та широкої популярності. В умовах конкуренції, розробники повинні постійно вдосконалювати свої пропозиції, щоб виділятися на тлі конкурентів. Регулярні оновлення, пакети розширень і дотримання нових тенденцій є життєво важливими для підтримки актуальності та залучення нових гравців. Зрештою, успіх залежить від створення захопливого ігрового процесу, який не лише відповідає, а й перевершує очікування сучасних гравців, забезпечуючи довготривалий вплив у яскравому світі багатокористувацьких партійних ігор.

1.4 Постановка задачі

Щоб задовольнити бажання людей, які шукають спільної взаємодії та спільного досвіду у віртуальному світі, необхідно створити ігровий додаток, що спеціалізується на multiplayer party games. Початкова ітерація складатиметься з трьох міні-ігор, доступних через головне меню: Turing Test, Brain Knights та Skibidy Party.

Turing Test – гра полягає в тому, щоб гравцям знайти бота серед реальних людей. Для полегшення орієнтування, ботів та людей, що відповідають на запитання, називають професорами, а людей, що вгадують та задають запитання, – студентами. Гра складається з 3-5-7-9 раундів в залежності від кількості гравців. У складі гравців може бути 4-8 осіб: у складі 4-5 гравців – 1-2 студенти, 3 професори; у складі 6 гравців – 2 студенти, 4 професори; у складі 7-8 гравців – 3 студенти, 4-5 професорів. Завдання студентів – набрати більше очок, ніж професори, задаючи

провокаційні питання та визначити після завершення раунду, хто з професорів є ботом, ґрунтуючись на їх відповідях. Завдання професорів – відповідати на питання студентів так, щоб сховатися під штучний інтелект, тобто надавати відповідь таким чином, як це зробив би бот. Якщо студенти вгадають бота, вони отримують два бали; якщо не вгадають, то професори отримують один бал.

Brain Knights – лицарі готують свої загіни, які складаються від 2 до 4 осіб, і виходять на арену турніру, переходячи одразу до його фіналу, де змагаються дві команди. Початковий раунд складається з двох підраундів і триває 90 секунд, включаючи 1,5 секунди на демонстрацію кожної правильної відповіді. Наступний раунд охоплює три окремі запитання. Він триває 141 секунду, з яких 40 секунд відводиться на надання кожної відповіді, а 7 секунд – на презентацію правильної відповіді. Кожен раунд має різний рівень складності запитань. Перший розрахований на бліц-опитування з простими запитаннями, де капітан обирає гравця зі своєї команди, який найкраще орієнтується у темі. У другому, навпаки, застосовується колективний підхід, що дозволяє командам спільно обмірковувати відповіді на більш складні запитання, де останнє слово залишається за капітаном. У кінці система підраховує бали в загальному підсумку, і загін, який набрав найбільшу кількість балів, отримує честь вступити до королівської гвардії. Міні-гра розрахована від 4 до 8 гравців.

Skibidy Party – гра має прості правила: є дві колоди карт – «меми» та «ситуації». Кожному учаснику роздають по шість карток із актуальними мемами. Потім по черзі кожен гравець стає суддею, який витягує карту «ситуації». Інші гравці, які не являються суддею, повинні вибрати зі своєї руки найсмішніший мем відповідно до «ситуації». Після цього, суддя обирає найсмішніші комбінації із запропонованих варіантів, даючи 3 призових місця гравцям. Автори найкращих підібраних комбінацій отримують бали, відповідно до місця, який надав суддя. Після цього, система підраховує усі зароблені бали, гравець з найбільшою їх кількістю перемагає та отримує звання «короля вечірки Skibidy». Міні-гра розрахована від 4 до 8 гравців.

У межах кваліфікаційної роботи на клієнтській частині необхідно розробити основне відображення геймплея на мобільний контролер, де він в свою чергу відповідає за надання команд гравця до системи, додавання тексту, зміну персонажу, відправку даних до сервера.

Також для забезпечення успішної гри в міні-іграх, ключовим є збалансованість рівня складності та ефективного розподілу часу. Стандартні психологічні теорії, такі як теорія потоку Міхалі Чіксентміхалі, підкреслюють важливість нерівномірного зростання складності гри, щоб забезпечити баланс між викликом і задоволенням [3]. Такий підхід дозволяє гравцям перебувати у стані привабливого потоку, де нерегулярне збільшення складності сприяє появі як невдач, так і успіхів, забезпечуючи цікавий та задоволений ігровий досвід.

Щоб забезпечити баланс у міні-грі Turing Test, дуже важливо ефективно розподілити час між складністю поставлених запитань, відповідями викладачів і подальшим аналізом, який виконують студенти. Такий баланс гарантує, що гра залишається цікавою і складною для всіх учасників, незалежно від їхнього рівня знань і знайомства з предметом [4].

Підтримувати баланс у міні-грі Brain Knights, дуже важливо ретельно відкалібрувати рівень складності кожного раунду. Це гарантує, що гравці матимуть достатньо часу для надання відповідей, але при цьому зіткнуться з відповідним рівнем складності. Крім того, дуже важливо збалансувати процес вибору респондента та теми для кожного раунду. Це дозволяє проводити стратегічні обговорення між членами команди, гарантуючи, що для відповіді на кожне запитання буде обрано найбільш підходящого гравця, а команда колективно обиратиме найбільш підходящу тему.

Для забезпечення балансу у міні-грі Skibidy Party, важливо ретельно продумати таймери, щоб надати гравцям і суддям достатньо часу для прийняття рішень. Це дозволяє вдумливо підбирати меми та ситуації, сприяючи загальній динаміці та задоволенню від гри. Крім того, введення елементу суддівства додає унікальної динаміки, впливаючи на рівень складності та створюючи можливості для стратегічного ігрового процесу.

Інструменти, що використовуються для балансу в іграх, включають в себе:

– таймери. Впровадження таймерів гарантує, що гравці мають обмежену кількість часу для прийняття рішень або виконання завдань, сприяючи темпу та перебігу гри;

– рівні складності. Регулювання рівнів складності завдань чи запитань гарантує, що гравці з різним рівнем навичок отримують відповідні виклики, що сприяє залученню та задоволенню;

– системи оцінювання. Системи підрахунку балів надають гравцям зворотній зв’язок щодо їхньої діяльності та прогресу, стимулюючи прийняття стратегічних рішень та ігровий процес;

– динамічні ігрові механіки. Включення динамічних ігрових механізмів, таких як змінна складність питань або рандомізовані події, робить гру непередбачуваною та цікавою для гравців [5].

Для розробки використовуємо середовище розробки Visual Studio Code у поєднанні з мовою програмування JavaScript [6] та бібліотекою React [7] для клієнтського веб-застосунку, а також TypeScript для серверної частини. Для забезпечення реального часу обміну даними між front-end та back-end частинами буде використовуватись бібліотека WebSocket.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Основна мета програми – розробити повноцінну multiplayer party game, акцентуючи увагу на балансі та впровадженні мобільного ігрового контролера для створення унікального способу управління грою. Надаючи пріоритет цим елементам, програма має на меті забезпечити захопливий і приємний ігровий досвід для гравців усіх рівнів.

Ігровий програмний застосунок проєктного застосунку повинен вирішувати наступні завдання:

– розробка інструментів для зручного керування через мобільний телефон;

– створення приватної кімнати для кожної ігрової сесії;

– справедливий баланс відносно часу та складності;

– створення інтуїтивного та якісного інтерфейсу та ігрового оточення.

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

а) баланс:

1) система повинна забезпечувати збалансований ігровий процес у всіх міні-іграх, щоб задовольнити гравців з різним рівнем підготовки;

2) реалізовані таймери з регульованою тривалістю, щоб забезпечити відповідний темп і рівень складності;

3) створення рівнів складності або механізмів масштабування, щоб динамічно змінювати складність завдання залежно від результатів гравців;

4) інтегрування системи підрахунку очок, які точно відображають прогрес та успіх гравця в міні-грі;

б) контролер на front-end частині:

1) загальне:

– можливість вписати нікнейм гравця;

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

– можливість перепідключення до лобі за допомогою унікального коду;

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

– відображення гравця який створив лобі;

2) міні-гра Skibidy Party:

– можливість відзначити готовність та почати гру;

– обрати ігрову карту з переліку розданих;

– оцінити ігрові карти та обрати призові місця;

– відображення підрахованого рахунку та статистики;

– відображення стану паузи;

3) міні-гра Brain Knights:

– можливість обрати унікальний аватар для гравця;

– можливість змінити команду;

– можливість змінити капітана команди;

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

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

– можливість вибрати одну з чотирьох відповідей;

– можливість обрати тему для раунда методом виключення;

– перегляд інформації про користувача;

– перегляд стану паузи;

4) міні-гра Turing Test:

– можливість змінити команду;

– можливість вписати питання для штучного інтелекту якщо гравець у команді студентів;

– можливість вписати відповідь на питання якщо гравець у команді професорів;

– можливість проголосувати за найпідозрілішу, на думку гравця, відповідь;

– відображення результату гри;
– відображення рахунку.

Високий рівень уваги до цих аспектів дозволяє створювати унікальні та захоплюючі ігри для гравців будь-якого віку та ігрового досвіду. Вони забезпечують цікавий ігровий досвід, який задовольняє потреби різних гравців.

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).

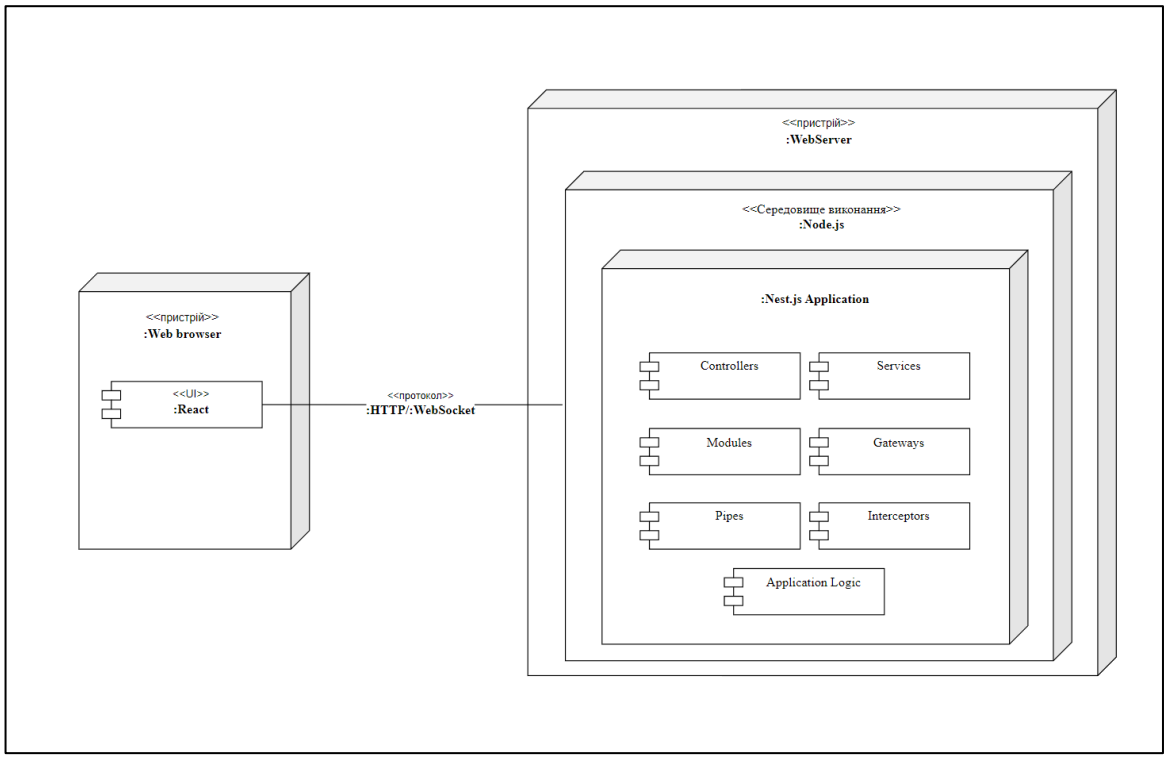


Рисунок 3.1 – UML діаграма розгортання ігрового програмного застосунку (діаграма виконана самостійно)

Ігровий програмний додаток складається з двох невід'ємних компонентів: веб-сервера та веб-додатку. Основною функцією веб-сервера є забезпечення стабільності мережі, обробка запитів на маршрутизацію, створення шлюзів веб-сокетів, а також управління обробкою, зберіганням та адмініструванням даних ігрової сесії. З іншого боку, веб-додаток пропонує користувачам зручний та інтуїтивно зрозумілий інтерфейс, доступний через веб-браузер. Він забезпечує комфортну взаємодію з системою, включаючи завдання управління, відображення інформації, передачу даних на сервер і обробку вхідних даних.

Для розроблюваного балансу ігрового програмного застосунку в жанрі multiplayer party games було створено діаграму прецедентів (див. рис. 3.2), діаграму діяльності (див. рис. 3.3), діаграму станів (див. рис. 3.4) та діаграму таймінгів (див. рис. 3.5-3.7).

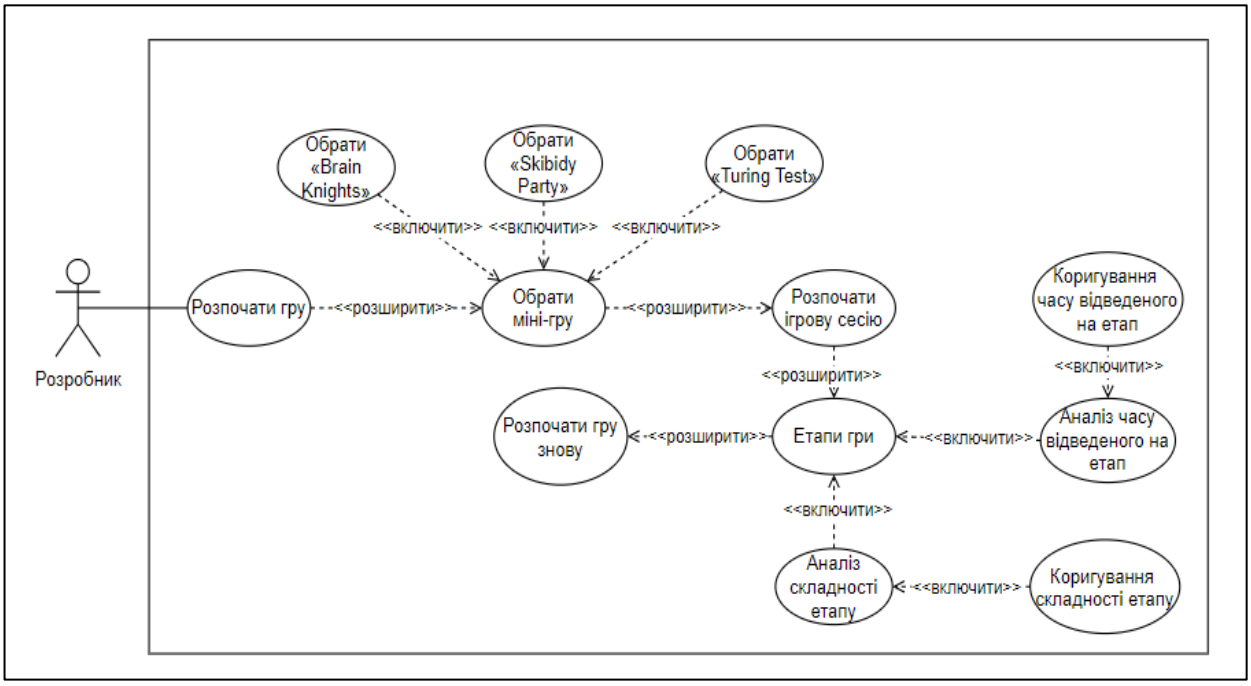


Рисунок 3.2 – UML діаграма прецедентів балансування ігрового програмного застосунку (діаграма виконана самостійно)

Розробник може взаємодіяти з системою наступним чином:

– обрати одну з трьох доступних міні-ігор;
– розпочинати ігрову сесію;
– розпочитати гру на різних етапах;
– аналізувати складність етапу;
– аналізувати час на етапі;
– коригувати складність етапу;
– коригувати час на етапі.

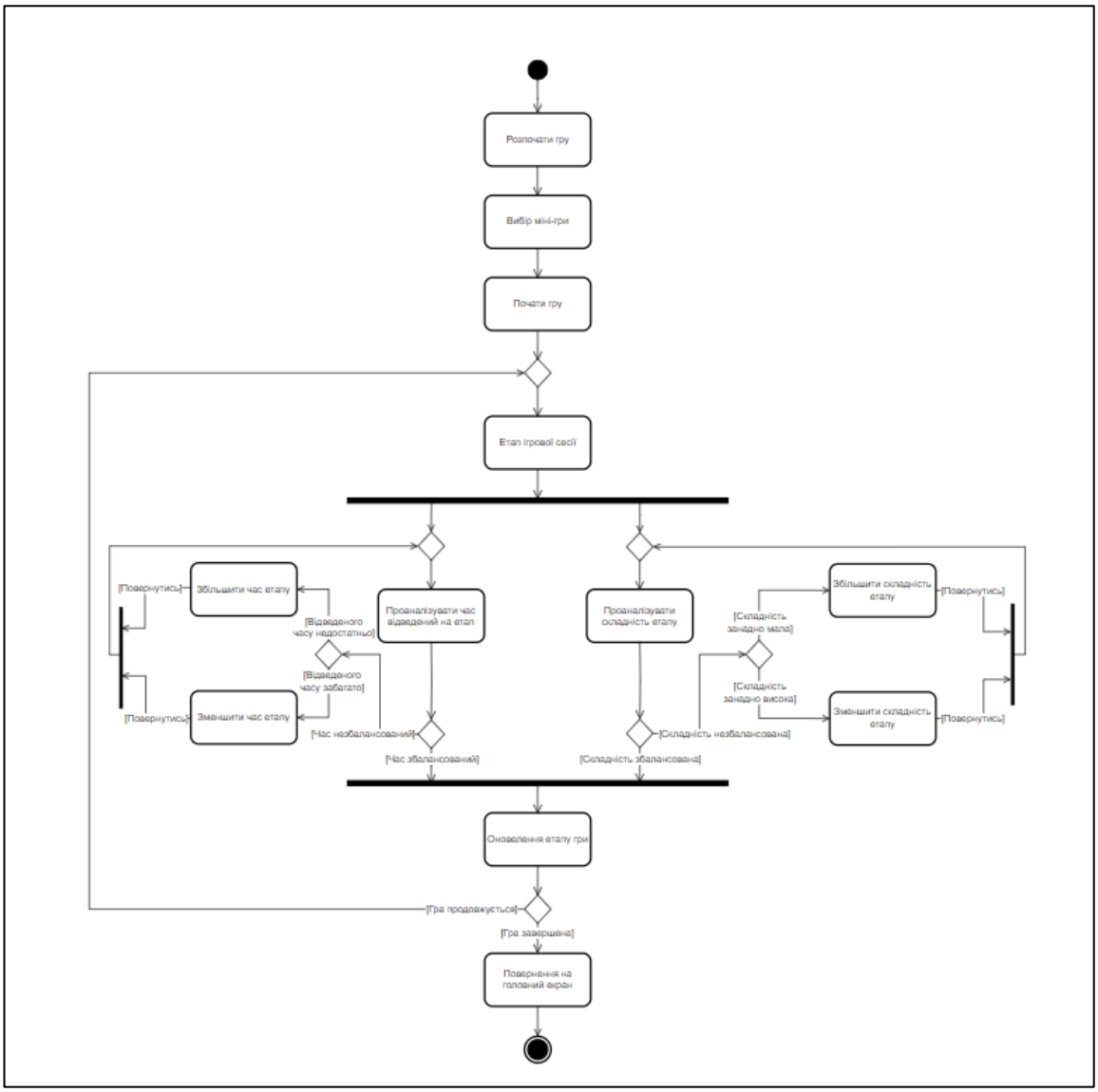


Рисунок 3.3 – UML діаграма діяльності балансування ігрового програмного застосунку (діаграма виконана самостійно)

На діаграмі діяльності, що відображає процес балансування, показано різні етапи міні-ігор. В рамках цієї схеми розробник ретельно аналізує можливості регулювання складності або тривалості, забезпечуючи оптимальний ігровий досвід.

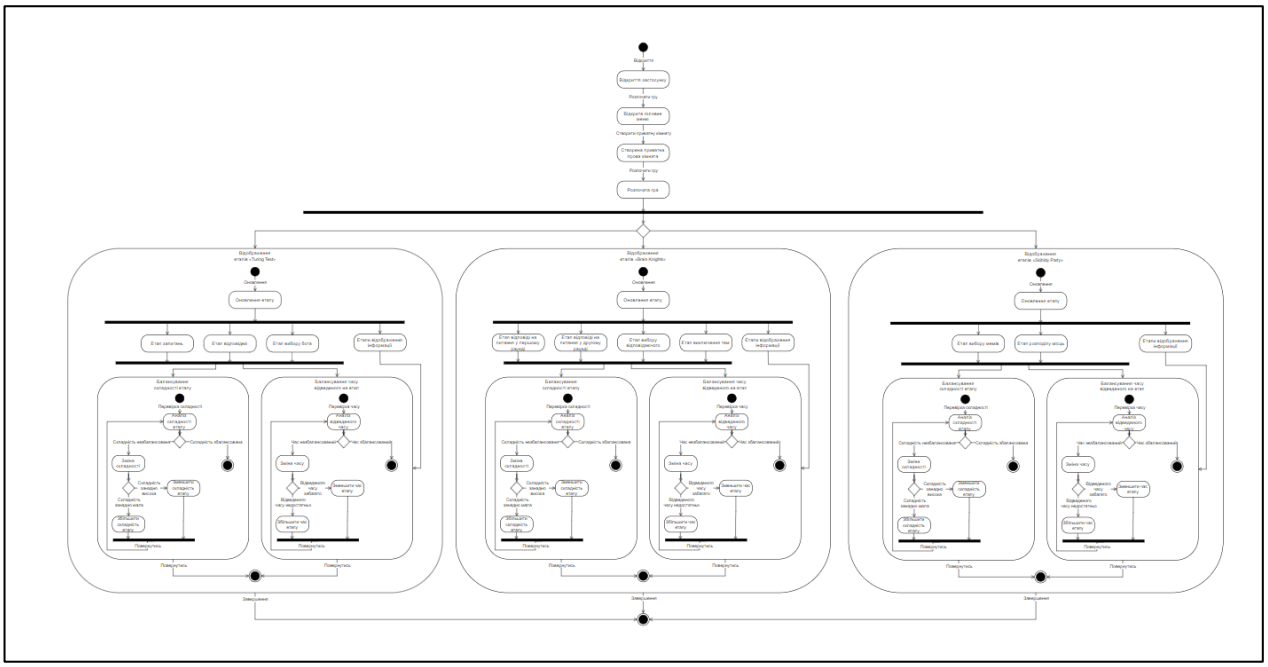


Рисунок 3.4 – UML діаграма станів балансування ігрового програмного застосунку (діаграма виконана самостійно)

На діаграмі станів, що відображає процес балансування, показано різні етапи міні-ігор. В рамках цієї схеми розробник ретельно аналізує можливості регулювання складності або тривалості, забезпечуючи оптимальний ігровий досвід [8].

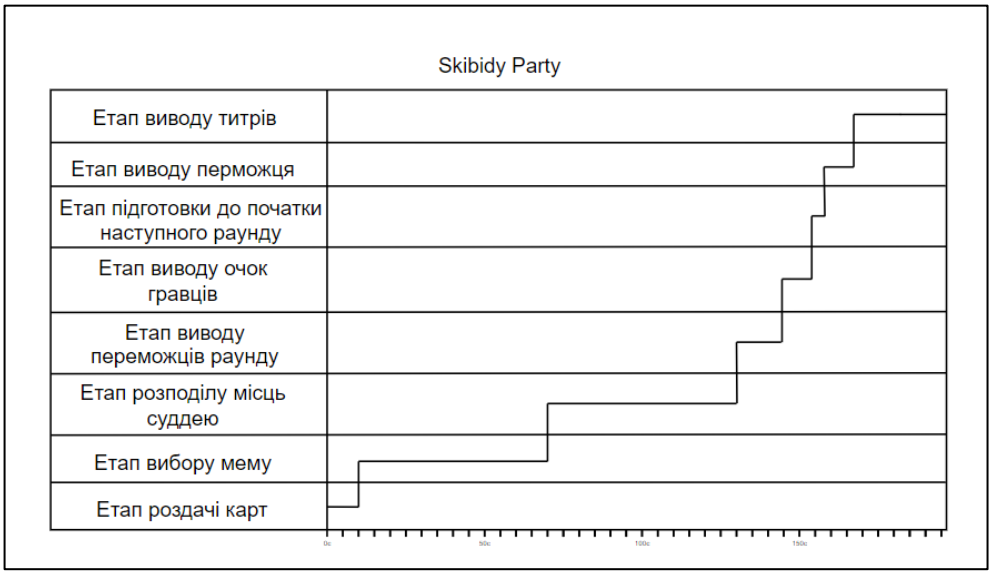


Рисунок 3.5 – UML діаграма таймінгів балансування ігрового програмного застосунку для гри Skibidy Party (діаграма виконана самостійно)

Діаграма таймінгів балансування для гри Skibidy Party відображає наданий час відповідно до етапів гри.

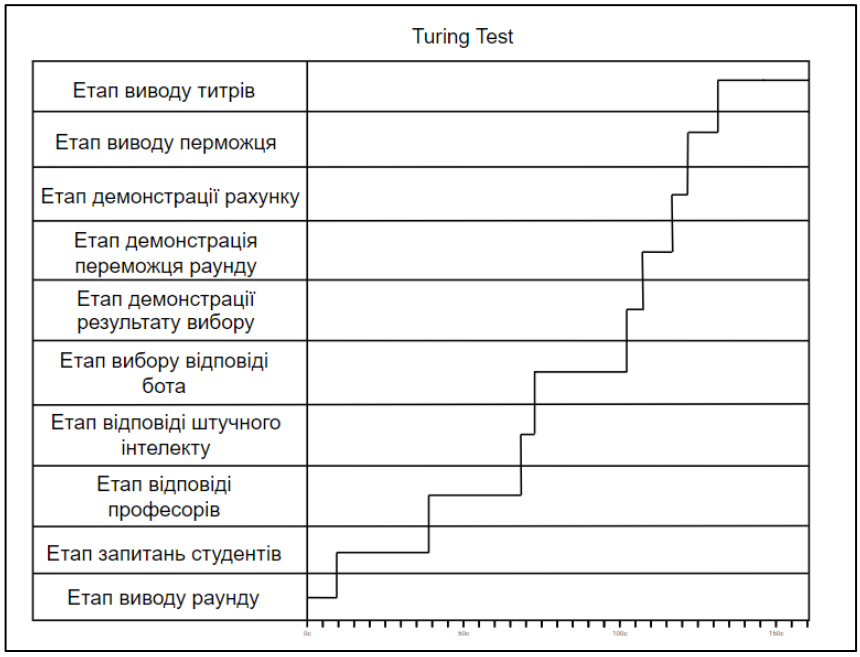


Рисунок 3.6 – UML діаграма таймінгів балансування ігрового програмного застосунку для гри Turing Test (діаграма виконана самостійно)

Діаграма таймінгів балансування для гри Turing Test відображає наданий час відповідно до етапів гри.

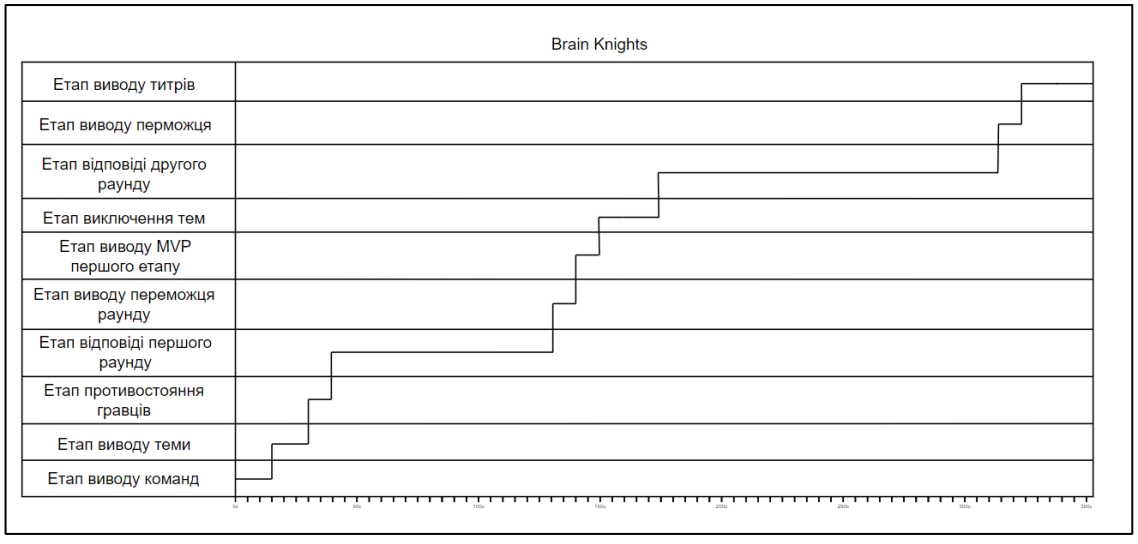


Рисунок 3.7 – UML діаграма таймінгів балансування ігрового програмного застосунку для гри Brain Knights (діаграма виконана самостійно)

Діаграма таймінгів балансування для гри Brain Knights відображає наданий час відповідно до етапів гри.

Для розроблюваного мобільного контролеру ігрового програмного застосунку в жанрі multiplayer party games було створено діаграму прецедентів (див. рис. 3.8), діаграму компонентів (див. рис. 3.9), діаграму пакетів (див. рис. 3.10), діаграму взаємодії (див. рис. 3.11), діаграму діяльності (див. рис. 3.12) та діаграму станів (див. рис. 3.13-3.16).

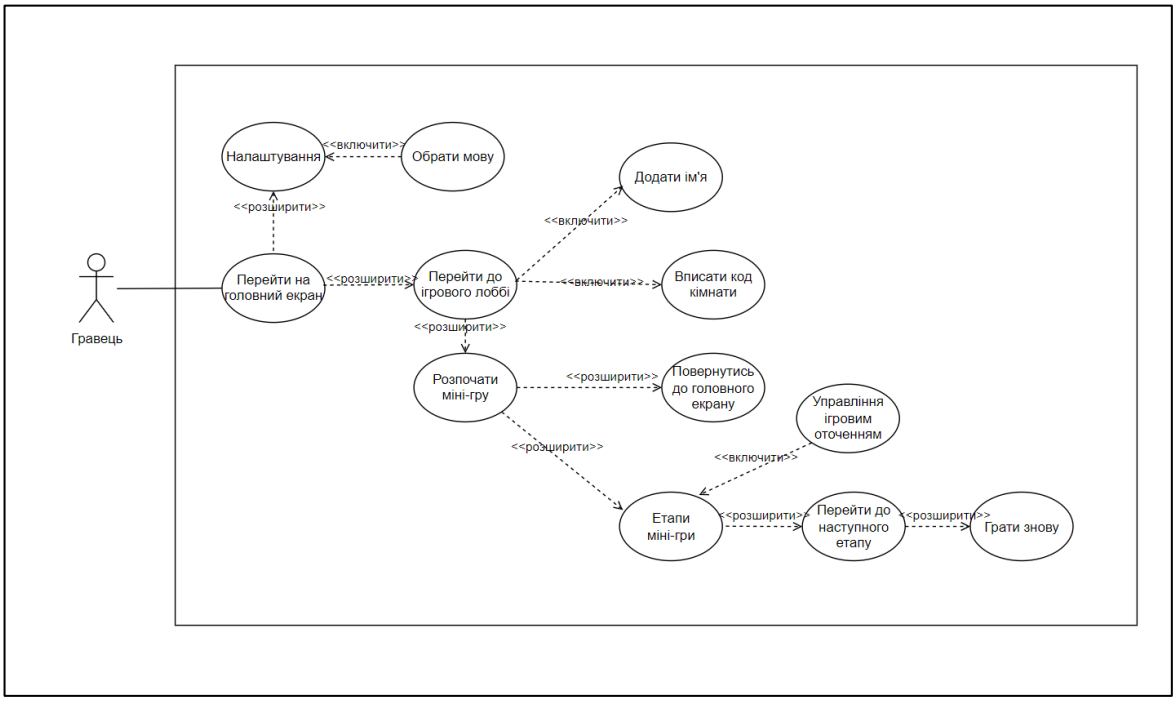


Рисунок 3.8 – UML діаграма прецедентів контролера на front-end частині ігрового програмного застосунку (діаграма виконана самостійно)

Користувач може взаємодіяти з системою наступним чином:

– налаштовувати локалізацію;
– додавати своє ім’я;
– підключитися до ігрової кімнати;
– додавати код кімнати;
– керувати ігровим процесом.

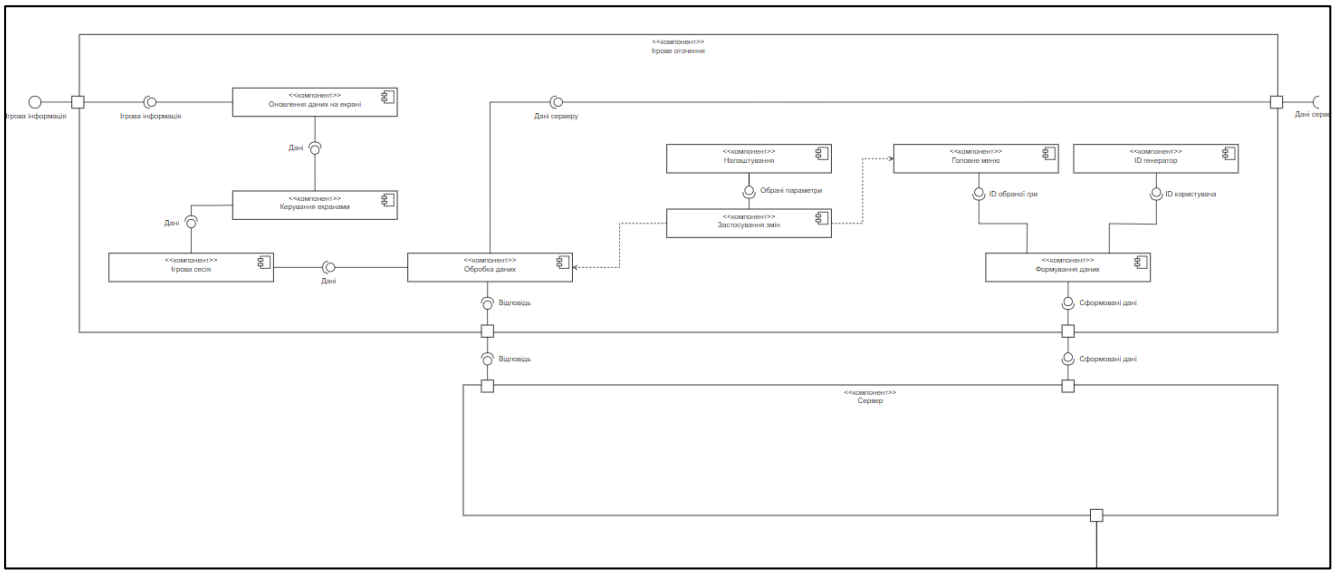


Рисунок 3.9 –UML діаграма компонентів контролера на front-end частині ігрового програмного застосунку (діаграма виконана самостійно)

Діаграма компонентів front-end частини демонструє взаємозв’язок між мобільним браузерним контролером, сервером програмного застосунку та середовищем. Вона описує, як мобільний ігровий контролер взаємодіє з сервером, керує ігровим процесом користувача та забезпечує механіку підключення до ігрової сесії.

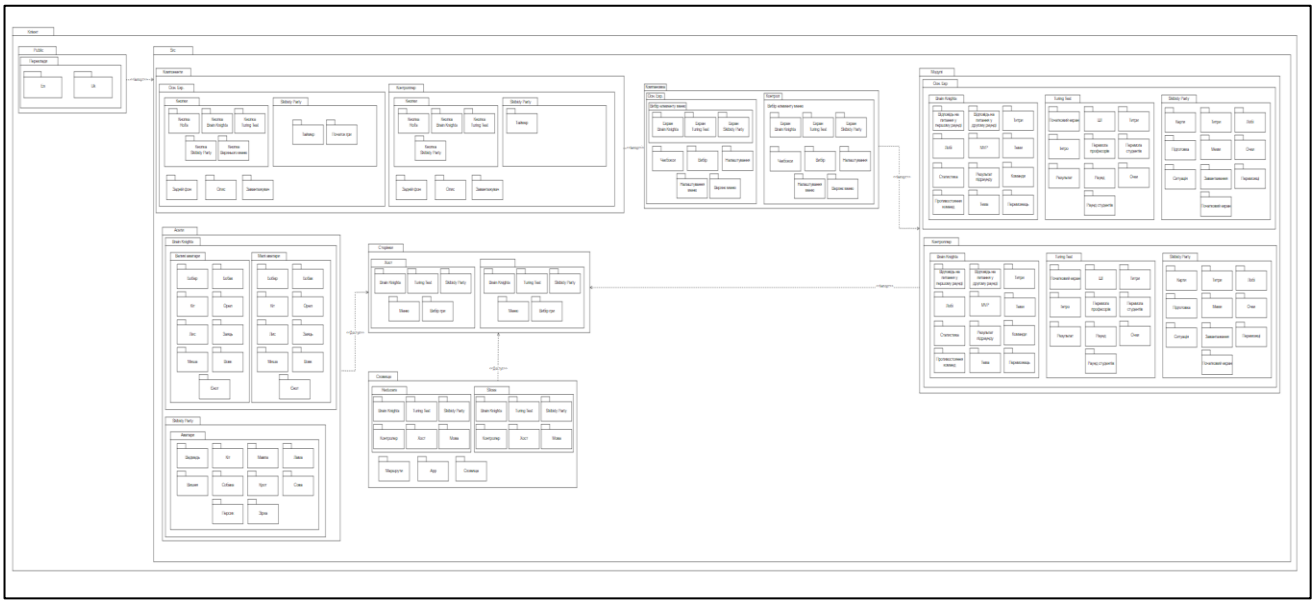


Рисунок 3.10 – UML діаграма пакетів front-end частини ігрового програмного застосунку (діаграма виконана самостійно)

Діаграма пакетів front-end частини показує організацію та розміщення елементів і модулів у проєкті. Вона ілюструє, як структуровані дані в ігровому інтерфейсі та мобільному контролері, які програмні ресурси розділені й ізольовані, а також які модулі використовують їх спільно.

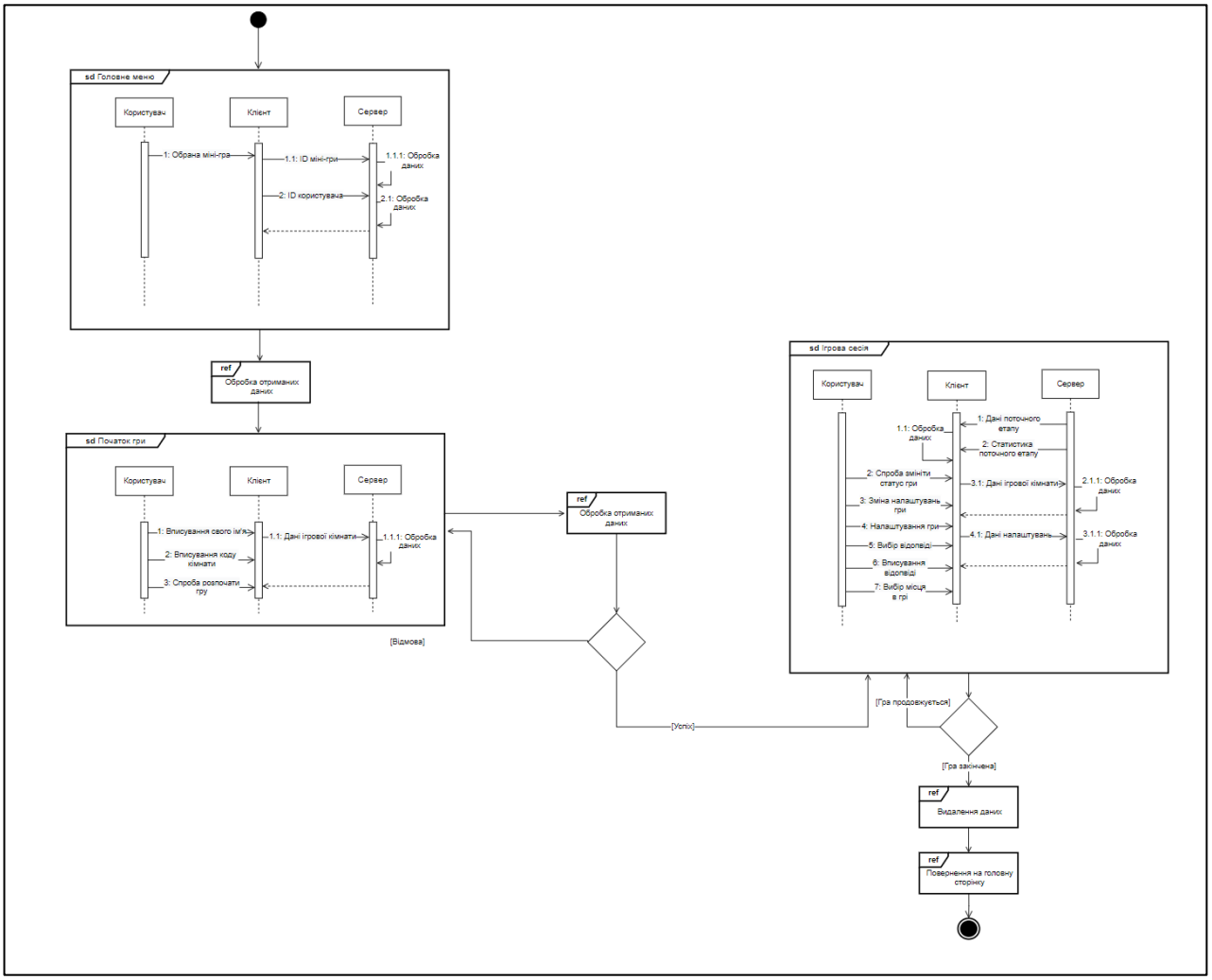


Рисунок 3.11 – UML діаграма взаємодії на front-end частині ігрового програмного застосунку (діаграма виконана самостійно)

Діаграма взаємодії у front-end частині наводить зображення потоку повідомлень всередині системи, де показана взаємодія між користувачами та компонентами цієї системи.

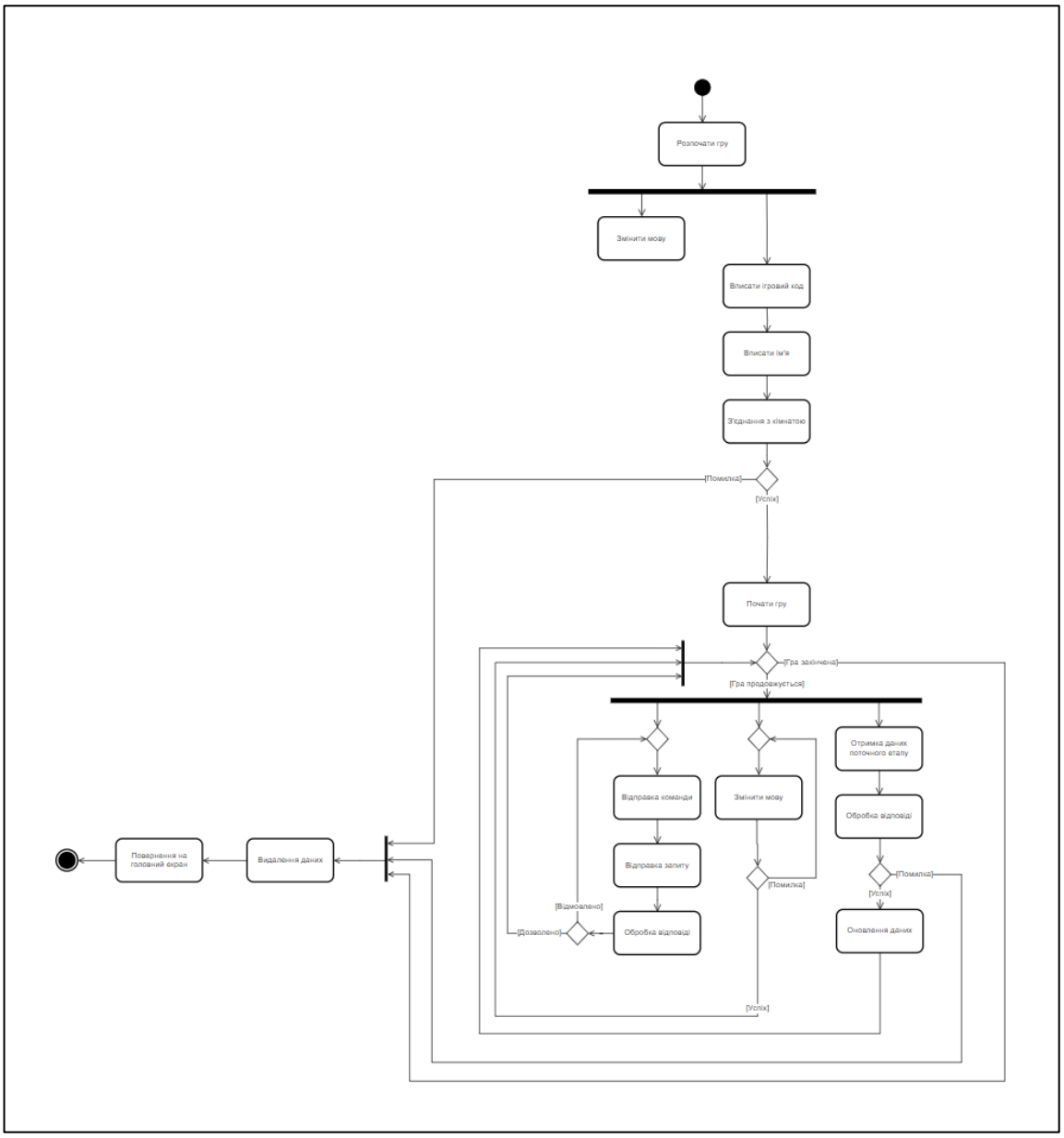


Рисунок 3.12 – UML діаграма діяльності контролера на front-end частині ігрового програмного застосунку (діаграма виконана самостійно)

UML діаграма діяльності контролера на front-end частини показує послідовну реалізацію логіки системи, де дії в застосунку переходять від однієї до іншої у цій конкретній частині системи.

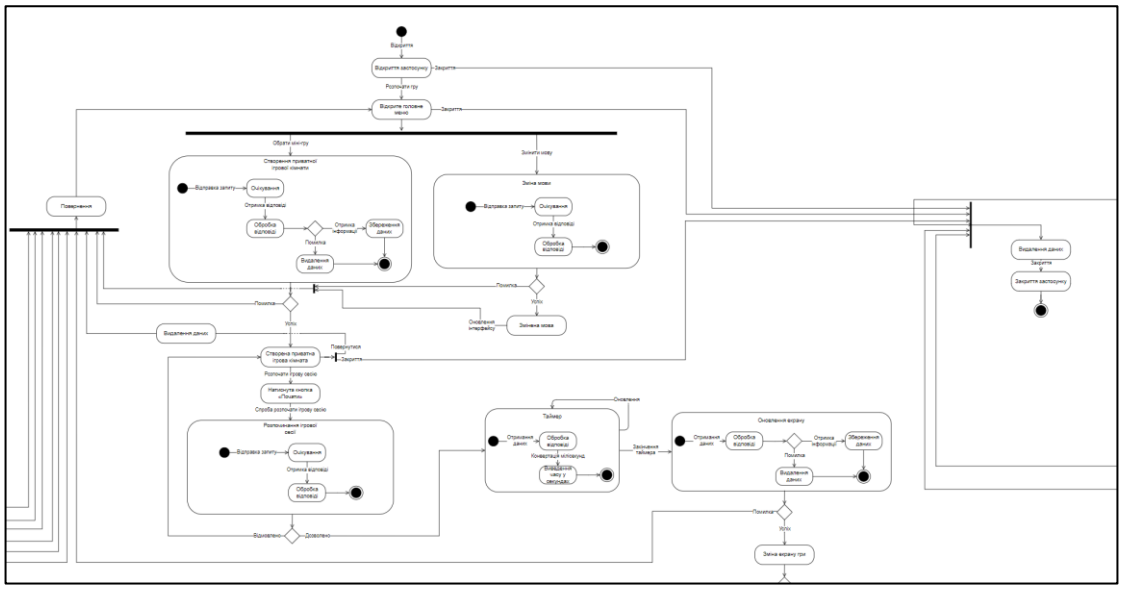


Рисунок 3.13 – UML діаграма станів контролера на front-end частині ігрового програмного застосунку, початок ігрової сесії (діаграма виконана самостійно)

У першій частині UML діаграми описується обробка логіки початку ігрової сесії на контролері. Ця частина включає в себе дії, такі як введення ігрового імені користувачем, створення коду кімнати, входження у кімнату, налаштування таймера та зміна локалізації.

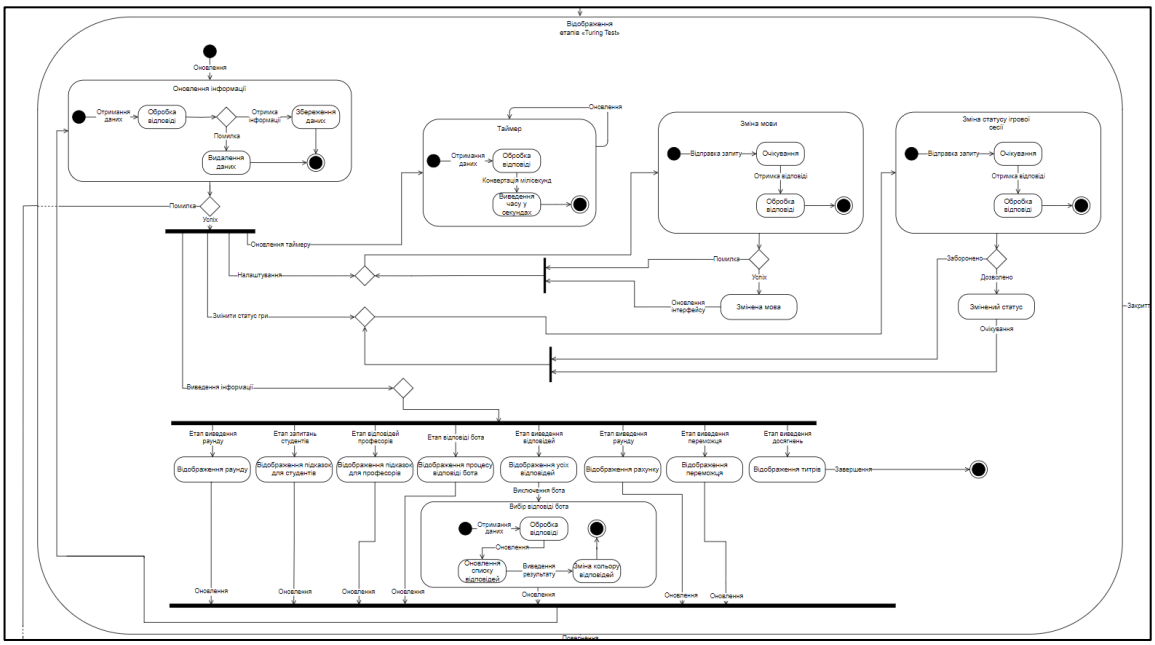


Рисунок 3.14 – UML діаграма станів контролера на front-end частині ігрового програмного застосунку. Міні-гра Turing Test (діаграма виконана самостійно)

У другій частині UML діаграми розглядається обробка логіки гри Turing Test на контролері. Це включає відображення таймера, збереження та видалення ігрових даних, зміну мови, додавання запитань та відповідей, відображення обробки відповідей штучного інтелекту, паузу гри, відображення поточного раунду та визначення переможця.

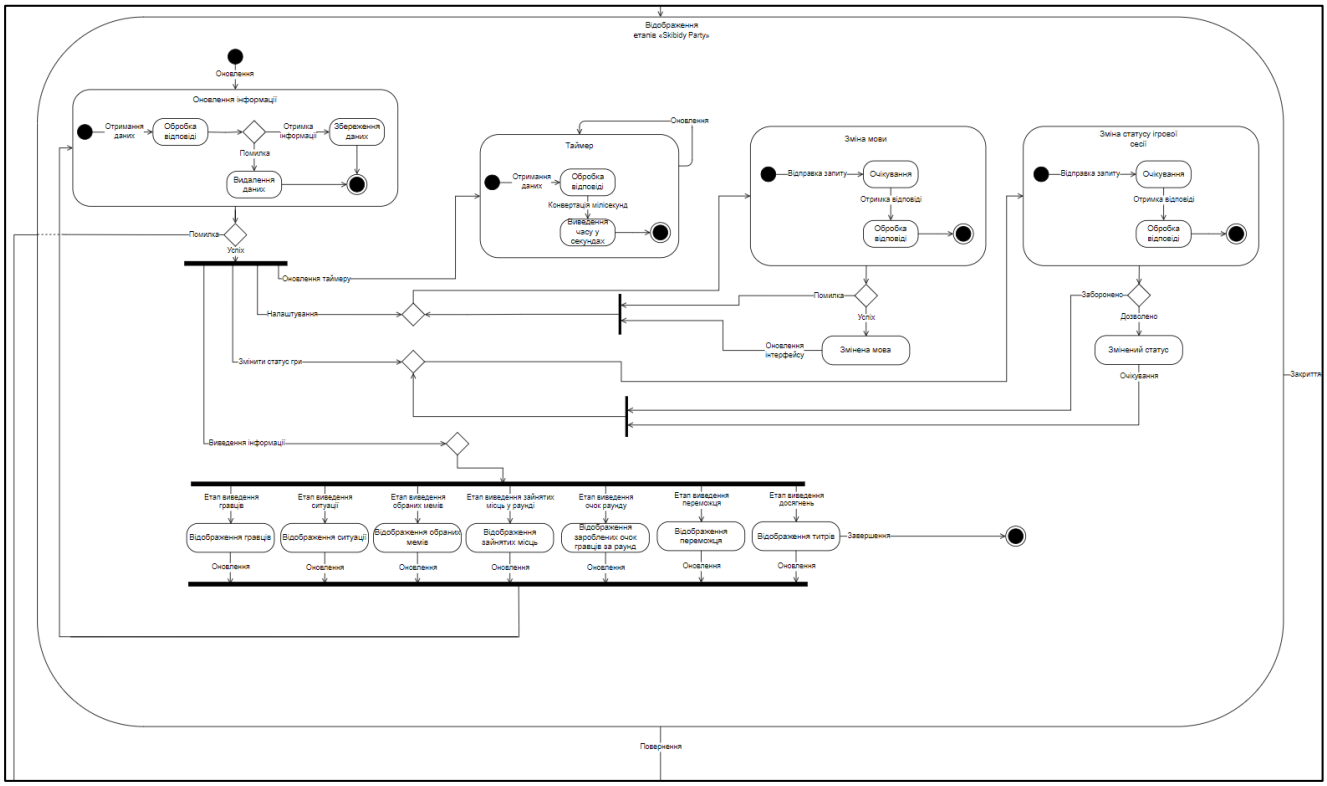


Рисунок 3.15 – UML діаграма станів контролера на front-end частині ігрового програмного застосунку. Міні-гра Skibidy Party (діаграма виконана самостійно)

У третій частині UML діаграми розглядається обробка логіки гри Skibidy Party на контролері. Це включає відображення таймера, збереження та видалення ігрових даних, показ мемів, вибір мемів для голосування, визначення кращого мему гравцем у ролі судді, відображення обраних мемів, пауза у грі та статистика раунду.

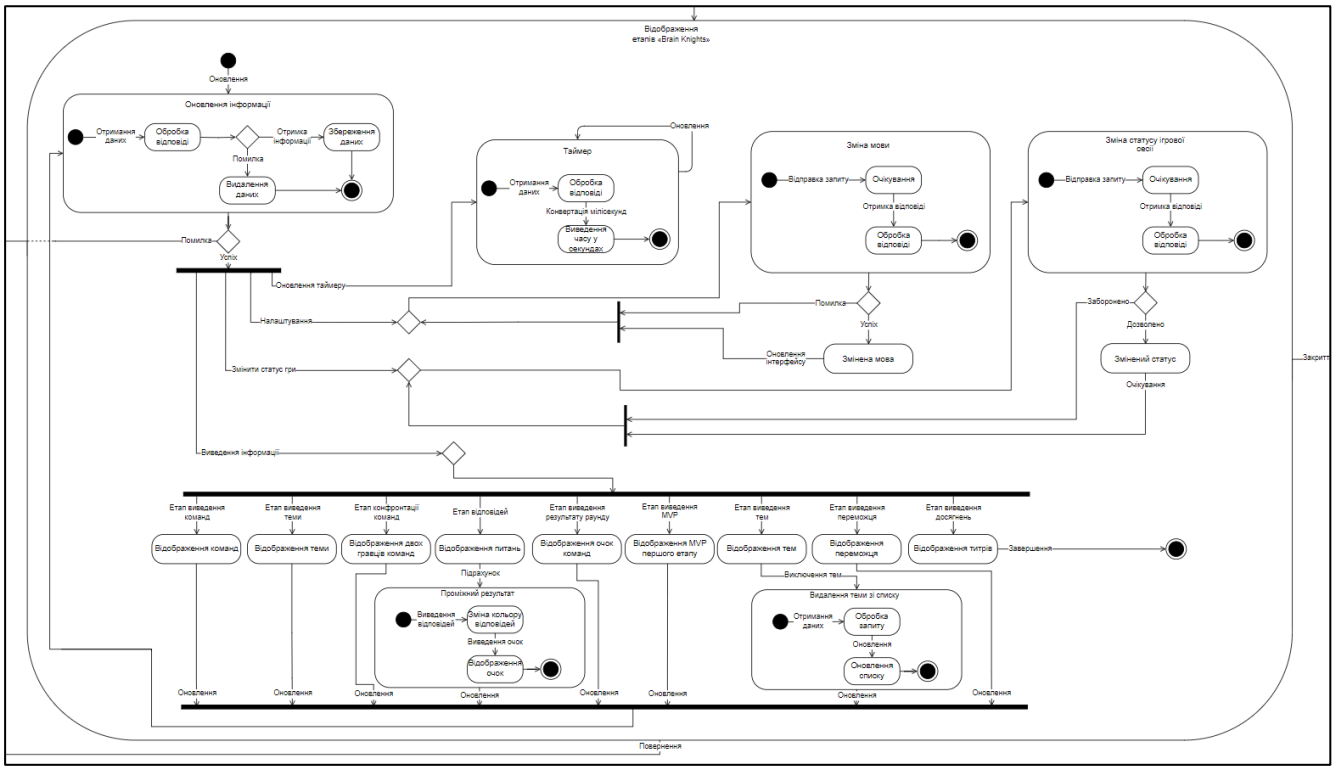


Рисунок 3.16 – UML діаграма станів контролера на front-end частині ігрового програмного застосунку. Міні-гра Brain Knights (діаграма виконана самостійно)

У четвертій частині UML діаграми подається обробка логіки гри Brain Knights. Зокрема, вона включає додавання назви команди, зміну команди гравцем, визначення капітана, вибір теми, обрання гравця для раунду, зміну мови, відображення питань і тем, визначення найкращого гравця раунду, а також оголошення переможців раунду та загального переможця гри, вибір відповіді, підкреслення правильних і неправильних відповідей.

3.2 Проєктування архітектури ПЗ

Front-end програмної системи використовує класичну односторінкову архітектуру (SPA) [6], що розділена на три рівні для ефективної організації функціональності (див. рис. 3.17). 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, що забезпечує централізоване управління станом. На третьому рівні здійснюється взаємодія з сервером через REST API для ефективного обміну даними.

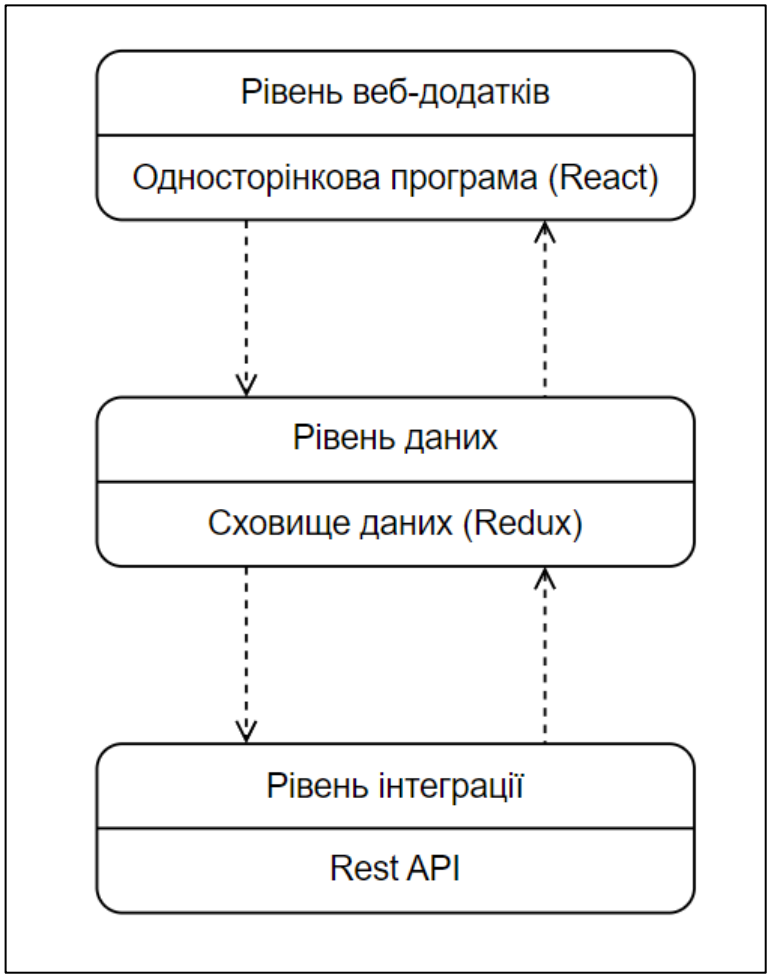


Рисунок 3.17 – Архітектура front-end частини проєкту (знімок екрана виконано самостійно)

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 front-end частини системи було спроєктовано наступну структуру (див. рис. 3.18):

– «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.

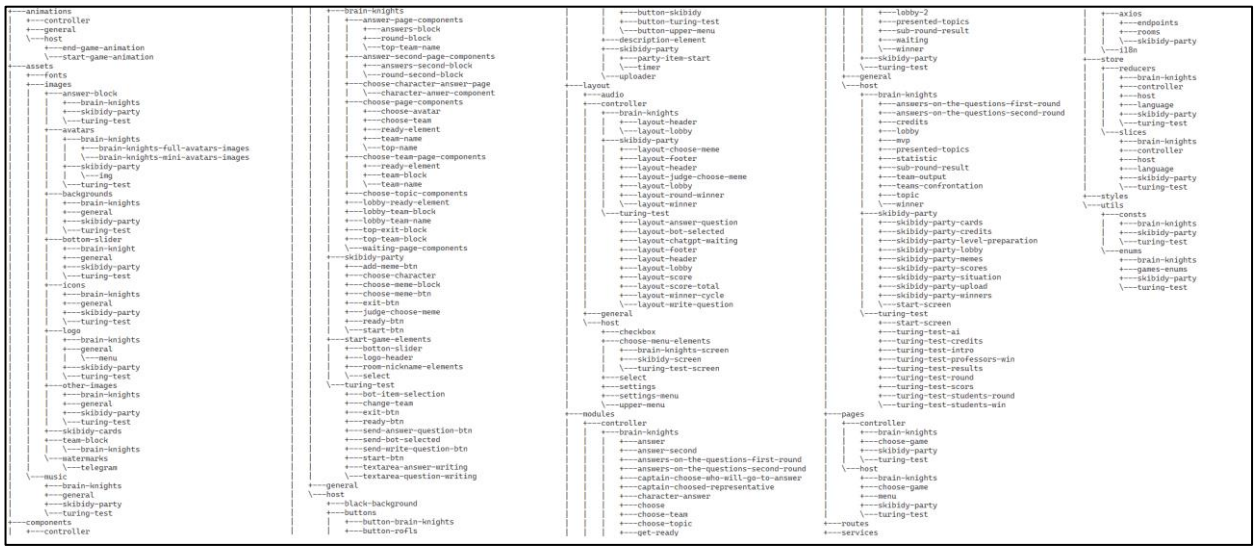


Рисунок 3.18 – Структура front-end частини проєкту (знімок екрана виконано самостійно)

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

3.3 Приклади найцікавіших алгоритмів та методів

Одним з ключових алгоритмів, є механізм динамічного регулювання складності, натхненний теорією потоків Міхала Чіксентміхалі. Цей алгоритм гарантує, що рівень складності гри коливається хвилеподібно, надаючи гравцям поєднання легших і складніших моментів. Уникаючи монотонності та запроваджуючи нерегулярне підвищення складності, гравці мають більше шансів пережити як невдачі, так і успіхи, що призводить до більшого задоволення та занурення у гру. До того, був розроблений алгоритм балансування ігрових раундів у кожній міні-грі. Ці алгоритми враховують такі фактори, як складність питання, тривалість раунду та взаємодію гравців, щоб забезпечити чесний та цікавий ігровий процес. Наприклад, у міні-грі Turing Test був збалансований час, відведений на формулювання запитань, надання відповідей та аналіз відповідей, щоб забезпечити справедливість і виклик. Використовування методів управління таймерами та різноманітністю завдань, посилюють конкуренцію та зберігають цілісність гри.

Загалом, дослідження підкреслює важливість системного підходу до таймінгу, складності та різноманітності завдань у multiplayer party games.

3.4 Створення UI / UX або іншого дизайну системи

Під час проєктування інтерфейсу був розроблений UI макет ігрових екранів контролеру, та в подальшому реалізований в самому ігровому застосунку [9].

На рисунку 3.19 ілюструється інтерфейс стартового екрану контролера, що дозволяє користувачам увійти до ігрової кімнати. За допомогою полів, гравці можуть ввести код кімнати та свої імена.

Рисунок 3.19 – Інтерфейс стартового екрану контролеру (знімок екрана виконано самостійно)

На рисунку 3.20 зображений інтерфейс вибору аватарів. Також маємо дві кнопки: «Ready» відповідає за зміну статусу гравця для початку гри, «Exit» відповідає за вихід з гри.

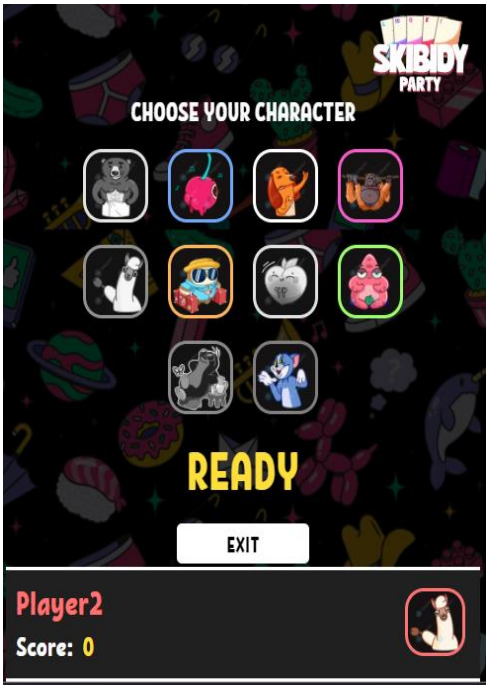


Рисунок 3.20 – Вибір аватару в Skibidy Party на контролері (знімок екрана виконано самостійно)

На рисунку 3.21 зображений інтерфейс вибору мему, де гравець обирає мем для ситуації, а потім додає його завдяки кнопці «Add meme».

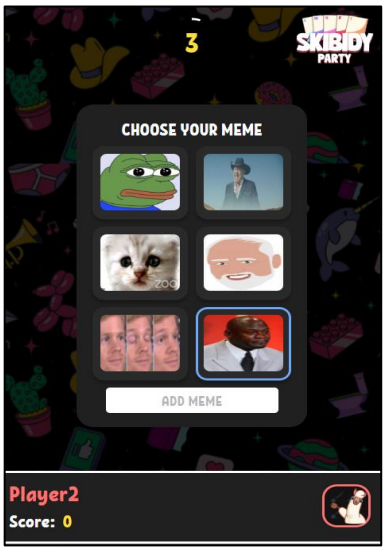


Рисунок 3.21 – Вибір мему в Skibidy Party на контролері (знімок екрана виконано самостійно)

На рисунку 3.22 зображений інтерфейс вибору переможця серед мемів суддею, Після розташування трьох призових місць, судді потрібно натиснути кнопку «Confirm».

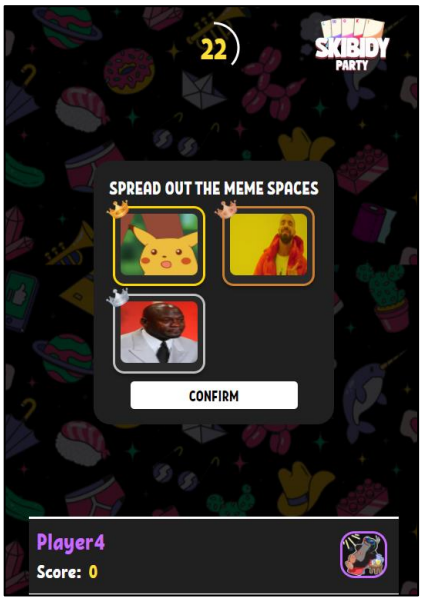


Рисунок 3.22 – Вибір переможного мему в Skibidy Party на контролері (знімок екрана виконано самостійно)

На рисунку 3.23 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.

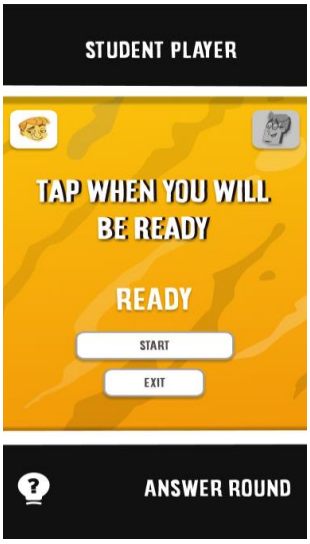


Рисунок 3.23 – Інтерфейс вибору команди в грі Turing Test на котролері (знімок екрана виконано самостійно)

На рисунку 3.24 зображений інтерфейс написання студентом питання. Для відправки питання – натиснути «Send».

Рисунок 3.24 – Інтерфейс написання питання студентами в грі Turing Test на котролері (знімок екрана виконано самостійно)

На рисунку 3.25 зображений інтерфейс написання відповіді на питання професорами. Для відправки відповіді – натиснути «Send».

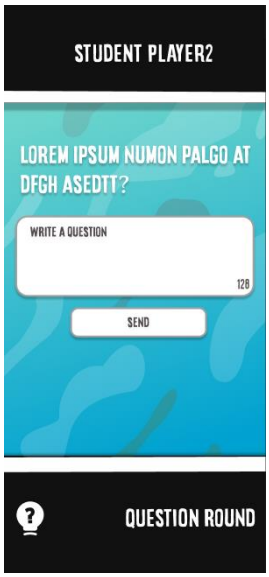


Рисунок 3.25 – Інтерфейс написання відповіді на питання професорами в грі Turing Test на котролері (знімок екрана виконано самостійно)

На рисунку 3.26 зображений інтерфейс вибору відповіді, де студенти аналізуючи, обирають відповідь ШІ. Після вибору номера відповіді для фінального голосування студенти повинні натиснути на кнопку «Send».

Рисунок 3.26 – Інтерфейс вибору відповіді ШІ в грі Turing Test на котролері (знімок екрана виконано самостійно)

На рисунку 3.27 зображений інтерфейс рахунку кожної ітерації раундів та в кінці гри.

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

На рисунку 3.28 зображено вибір ігрового аватару при вході до ігрової кімнати.

Рисунок 3.28 – Вибір ігрового аватара в грі Brain Knights на хості (знімок екрана виконано самостійно)

На рисунку 3.29 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.

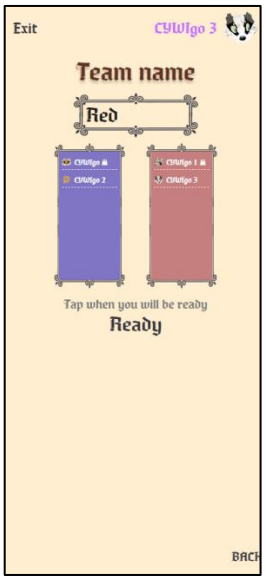


Рисунок 3.29 – Вибір команди в грі Brain Knights на хості (знімок екрана виконано самостійно)

На рисунку 3.30 зображений вибір гравця для участі у бліц-опитуванні.

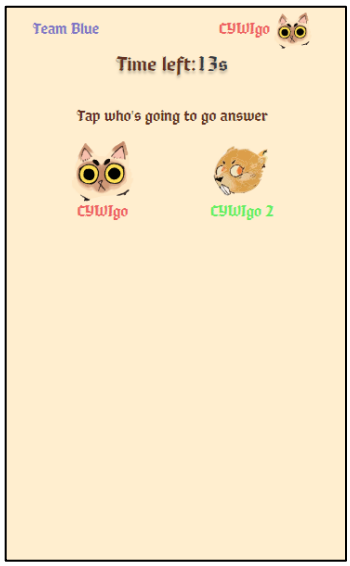


Рисунок 3.30 – Вибір гравця для відповіді в грі Brain Knights на хості (знімок екрана виконано самостійно)

На рисунку 3.31 зображений вибір відповіді гравцем на запитання, на екрані також зроблений таймер, який показує залишок часу до кінця раунду.

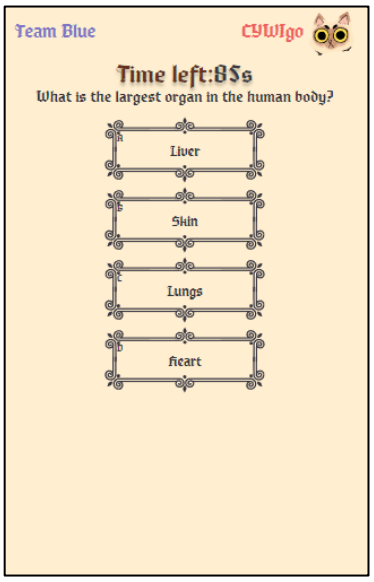


Рисунок 3.31 – Вибір відповіді на запитання в грі Brain Knights на хості (знімок екрана виконано самостійно)

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.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

Front-end частина ігрового програмного застосунку у жанрі multiplayer party games була створена за допомогою мови програмування JavaScript, платформи Node.js [10] та фреймворку React. Нижче наведені основні бібліотеки та інструменти, які використовувалися у цьому проекті:

– @reduxjs/toolkit. Спрощує роботу з Redux, надаючи інструменти для створення та керування станом додатку;

– @testing-library/react. Інструменти для тестування React-компонентів, які забезпечують кращі практики тестування;

– @testing-library/user-event. Утиліти для імітації дій користувача (кліки, набір тексту) у тестах;

– axios. Бібліотека для виконання HTTP-запитів, підтримує проміси та асинхронні операції;

– framer-motion. Бібліотека для створення анімацій та жестів у React-додатках;

– i18next. Бібліотека для інтернаціоналізації додатків, підтримує багато мов та форматів;

– i18next-browser-languagedetector. Розширення для i18next, яке автоматично визначає мову користувача;

– js-cookie. Утиліта для роботи з куками (cookies) у браузері;

– react. Основна бібліотека для створення користувацьких інтерфейсів на основі компонентів;

– react-dom. Пакет для роботи з DOM у React-додатках;

– react-qr-code. Компонент для генерації QR-кодів у React-додатках;

– react-redux. Інтеграція Redux з React, надає зв'язуючий код для роботи зі станом додатку;

– react-responsive-carousel. Компонент для створення каруселей та слайдерів у React-додатках;

– react-router-dom. Бібліотека для маршрутизації у React-додатках, дозволяє створювати навігаційні маршрути;

– sass. Препроцесор CSS, що додає можливості використання змінних, вкладеності та інших розширень;

– socket.io-client. Клієнт для роботи з WebSocket на боці браузера, використовується для створення реального часу з'єднань [11];

– uuidv4. Генератор UUID (унікальних ідентифікаторів).

Файл .env використовується для зберігання конфіденційних даних і конфігураційних параметрів, які можуть бути використані додатком під час виконання. Це дає змогу легко керувати та змінювати параметри конфігурації, не потребуючи модифікації коду. Давайте розглянемо конкретні параметри, які вказані у файлі .env:

– змінна PORT. Вказує номер порту, на якому буде запущено ваш сервер або додаток;

– змінна REACT_APP_BACKEND_URL. Вказує URL адресу сервера бекенда, до якого фронтенд робитиме запити.

Для початкового налаштування ідентифікатора контролера на основі інформації, збереженої в куках браузера використовується функція initializeId. Вона перевіряє наявність поточного ідентифікатора контролера у куках браузера. Якщо такий ідентифікатор існує, він використовується для стану програми. У випадку, якщо ідентифікатор відсутній, генерується новий за допомогою функції uuidv4(), який потім зберігається в куках для подальшого використання.

```
const initializeId = (state) => {
  const currentControllerId = Cookies.get("controllerId");
  if (currentControllerId) {
    state.controller.id = currentControllerId;
  } else {
    const id = uuidv4();
    state.controller.id = id;
    Cookies.set("controllerId", id);
  }
};
```

Функція handleSwitchTeamClick призначена для обробки події кліку на кнопку зміни команди в грі Brain Knights. Вона перевіряє статус гри та виконує дії лише у випадку, якщо гра перебуває в статусі очікування. Потім вона визначає

команду гравця та перевіряє, чи він вже належить до обраної команди. Якщо користувач є членом обраної команди, функція завершується без подальших дій. У протилежному випадку вона відправляє сигнал серверу про зміну команди, змінює назву команди користувача та з'єднується з сервером. Крім того, повертається функція, яка відключається від сервера після завершення виконання.

```
const handleSwitchTeamClick = teamId => {
      if (brainKnightsState.gameSession?.status === 'waiting') {
            const userTeam = brainKnightsGetTeam()
            const isUserFirstTeam =
                  userTeam === brainKnightsState.gameSession.firstTeam
            const isUserSecondTeam =
                  userTeam === brainKnightsState.gameSession.secondTeam
            if (
                  (teamId === 'red' && isUserFirstTeam) ||
                  (teamId === 'blue' && isUserSecondTeam)
            ) {
                  return
            }
            socket.emit('switchBrainKnightsControllerTeam', {
                  roomId: controllerState.lastSelectedRoom?.id,
                  controllerId: controllerState.controller?.id,
            })
            if (
                  isUserFirstTeam &&
                  controllerState.controller?.id ===

brainKnightsState.gameSession.firstTeam.captainId
            ) {

setNewTeamName(brainKnightsState.gameSession.secondTeam.name)
            } else if (
                  isUserSecondTeam &&
                  controllerState.controller?.id ===

brainKnightsState.gameSession.secondTeam.captainId
            ) {

setNewTeamName(brainKnightsState.gameSession.firstTeam.name)
            }
            socket.connect()
            return () => {
                  socket.disconnect()
                  socket.off('connect')
                  socket.off('switchBrainKnightsControllerTeam')
            }
      }
}
```

Функція gameResult визначає результат гри з урахуванням переможця або нічиєї у грі Brain Knights. Якщо обидві команди мають певну кількість очок (представлену властивістю length у gameWinner), то повертається повідомлення

про нічию. У випадку, коли одна з команд має більше очок, ніж інша, повертається повідомлення з назвою переможеної команди. Якщо жодна з команд не має балів, повертається порожній рядок.

```
const gameResult = () => {
    if (gameWinner.firstTeam?.length > 0 && gameWinner.secondTeam?.length > 0) {
        return <span style={{ color: '#8075C5' }}>{t('Both teams')}</span>
    } else if (
        gameWinner.firstTeam?.length > 0 &&
        gameWinner.secondTeam?.length === 0
    ) {
        return (
            <span style={{ color: '#C57F7F' }}>
                {brainKnightsState.gameSession?.firstTeam?.name} {t('Team')}
            </span>
        )
    } else if (
        gameWinner.firstTeam?.length === 0 &&
        gameWinner.secondTeam?.length > 0
    ) {
        return (
            <span style={{ color: '383741' }}>

{brainKnightsState.gameSession?.secondTeam?.name} {t('Team')}
            </span>
        )
    } else {
        return ''
    }
}
```

Функція brainKnightsTeamAffiliation визначає приналежність контролера до команди в грі Brain Knights. Вона перевіряє, чи є контролер капітаном однієї з команд. Якщо так, повертається label про те, що він є капітаном цієї команди. Якщо контролер не є капітаном жодної команди, то перевіряється, чи є він учасником однієї з команд. Якщо так, повертається повідомлення про приналежність до відповідної команди. Якщо ж контролер не є ні капітаном, ні учасником жодної з команд, повертається повідомлення про те, що він не належить до жодної команди.

```
const brainKnightsTeamAffiliation = () => {
    if (
        brainKnightsState.gameSession?.firstTeam?.captainId ===
        controllerState.controller?.id
    ) {
        return `${t('Captain in')} ${
            brainKnightsState.gameSession?.firstTeam?.name
        } ${t('Team')}`
    } else if (
```

```
                brainKnightsState.gameSession?.secondTeam?.captainId ===
                controllerState.controller?.id
          ) {
                return `${t('Captain in')} ${
                      brainKnightsState.gameSession?.secondTeam?.name
                } ${t('Team')}`
          } else if (

     brainKnightsState.gameSession?.firstTeam?.controllers?.some(
                      c => c?.id === controllerState.controller?.id
                )
          ) {
                return   `${brainKnightsState.gameSession?.firstTeam?.name}
${t('Team')}`
          } else if (

     brainKnightsState.gameSession?.secondTeam?.controllers?.some(
                      c => c?.id === controllerState.controller?.id
                )
          ) {
                return  `${brainKnightsState.gameSession?.secondTeam?.name}
${t('Team')}`
          } else {
                return `${t('No Team')}`
          }
     }
```

Функція updateRoundWinners оновлює переможців раунду на основі ситуації в грі. Вона використовується для визначення, які гравці отримали найвищий рейтинг у поточній ситуації гри. Крім того, вона враховує меми, які були роздані гравцям. Коли визначені переможці, їхні дані зберігаються у відповідній змінній та оновлюються.

```
     const updateRoundWinners = useCallback(room => {
          const gameSession = room.gameSession
          const controllers = room.controllers
          const situation = gameSession.situations.find(
                situation => situation.nowDisplayed && !situation.wasUsed
          )

          if (situation) {
                const situationMemesIds = situation.memes
                if (situationMemesIds.length > 0) {
                      const memesWithspots = situationMemesIds
                            .map(id => gameSession.memes.find(meme => meme.id
=== id))
                            .filter(meme => meme && meme.spot)
                      if (memesWithspots.length > 0) {
                            let tempRoundWinners = {
                                  1: null,
                                  2: null,
                                  3: null,
                            }
                            const sortedMemesWithspots = [...memesWithspots]
```

```
                        sortedMemesWithspots.sort((a, b) => b.spot -
a.spot)
                        sortedMemesWithspots.forEach(memeWithSpot => {
                              if (memeWithSpot.memeDealtTo) {
                                    const        foundController        =
controllers.find(
                                          item     =>     item.id     ===
memeWithSpot.memeDealtTo
                                    )
                                    const          existingSpot          =
Object.values(tempRoundWinners).find(
                                          controller   =>   controller   &&
controller.id === foundController.id
                                    )
                                    if (!existingSpot) {

      tempRoundWinners[memeWithSpot.spot] = foundController
                                    }
                              }
                        })
                        setRoundWinners(tempRoundWinners)
                  }
            }
      }
}, [])
```

Функція getWinners призначена для визначення переможців у грі Skibidy Party. Вона отримує список контролерів гри, сортує їх за загальними балами та визначає гравців з максимальними балами. Переможцями стають гравці з найвищими балами. Якщо декілька гравців мають однакову кількість балів, всі вони додаються до списку переможців.

```
const getWinners = () => {
      let sortedControllers = [...skibidyPartyState.controllers];
      const winners = [];

      if (sortedControllers.length > 0) {
        sortedControllers.sort((a, b) => b.scores - a.scores);

        let maxScores = sortedControllers[0].scores;
        winners.push(sortedControllers[0]);

        for (let i = 1; i < sortedControllers.length; i++) {
            if (sortedControllers[i].scores === maxScores) {
              winners.push(sortedControllers[i]);
            } else {
              break;
            }
        }
      }
```

Функція updateCurrentQuestionsForAnswer оновлює список поточних питань для відповідей на основі стану гри Turing Test. Вона використовується для

визначення питань, які повинні бути відповідані у поточному циклі гри. Функція отримує цикл гри та список всіх питань. На основі циклу вона відфільтровує питання, які належать до поточного циклу гри. Далі вона додає ім'я студента, який виніс питання, до кожного відфільтрованого питання. Оновлені питання зберігаються та оновлюються у стані програми для подальшого використання.

```
const updateCurrentQuestionsForAnswer = useCallback(room => {
      const cycle = room.gameSession.cycle
      const questions = room.gameSession.questions
      const updatedQuestions = questions
            .filter(item => item.cycle === cycle)
            .map(question => {
                  const student = room.controllers.find(
                        controller =>
                              controller.id === question.controllerId &&
                              controller.team === teamEnum.STUDENTS
                  )
                  if (student) {
                        return {
                              ...question,
                              controllerName: student.name,
                        }
                  }
                  return question
            })
      setCurrentQuestionsForAnswer(updatedQuestions)
}, [])
```

Функція changeLanguage призначена для зміни мови інтерфейсу в залежності від налаштувань гри. Вона викликається при зміні мови в грі та отримує інформацію про мову, встановлену для сесії гри та загальну мову. Якщо для гри встановлено загальну мову, функція змінює мову інтерфейсу на мову гри та оновлює її у стані програми. У протилежному випадку вона змінює мову інтерфейсу на особисту мову контролера гри та оновлює її у стані програми.

```
const changeLanguage = useCallback(
      room => {
            const language = room.gameSession.language
            const commonLanguage = room.gameSession.commonLanguage
            if (commonLanguage) {
                  i18n.changeLanguage(language)
                  dispatch(setLanguage(language))
            } else {

i18n.changeLanguage(languageState.personalControllerLanguage)

dispatch(setLanguage(languageState.personalControllerLanguage))
            }
      },
      [dispatch, i18n, languageState.personalControllerLanguage]
```

)

Під час впровадження функцій контролера на клієнтській стороні програмної гри, було забезпечено покриття всіх аспектів гри, включаючи логіку всіх трьох ігор. Це дозволило забезпечити коректне функціонування та оптимальний користувацький досвід у грі.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування повинне охоплювати всі реалізовані функції контролера на стороні клієнта в ігровому програмному додатку для кваліфікаційного проекту, включаючи логіку всіх трьох ігор.

Для цієї оцінки було використано ручне тестування. Метод забезпечує ретельну увагу до деталей та виявлення суттєвих проблем. Він дозволяє швидко виявляти та вирішувати помилки, а також пропонує гнучкість у застосуванні нестандартних методів тестування та інноваційних підходів.

Під час ручного тестування ігрового програмного забезпечення на відповідність усім критеріям, викладеним у технічному завданні, важливим є структурний підхід. Процес тестування на стороні клієнта включав наступні етапи:

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

б) перевірка функціональних вимог:

1) баланс:

– перевірено збалансований ігровий процес у всіх міні-іграх;

– перевірено реалізовані таймери з регульованою тривалістю;

– перевірено створення рівнів складності або механізмів масштабування;

– перевірено інтегрування системи підрахунку балів;

2) загальне:

– перевірено можливість вписати нікнейм гравця;

– перевірено можливість під’єднатися до ігрового лобі за допомогою унікального сгенерованого коду нікнейм гравця;

– перевірено можливість перепідключення до лобі за допомогою унікального коду;

– перевірено можливість поставити гру на паузу у будь-який момент гри;

– перевірено відображення гравця який створив лобі;

3) міні-гра Skibidy Party:

– перевірено можливість відзначити готовність та почати гру;

– перевірено обрати ігрову карту з переліку розданих;

– перевірено оцінити ігрові карти та обрати призові місця;

– перевірено відображення підрахованого рахунку та статистики;

– перевірено відображення стану паузи;

4) міні-гра Brain Knights:

– перевірено можливість обрати унікальний аватар для гравця;

– перевірено можливість змінити команду;

– перевірено можливість змінити капітана команди;

– перевірено можливість обрати назву для команди якщо гравець грає у ролі капітана;

– перевірено можливість капітану обрати гравця який буде відповідати у бліц-раунді;

– перевірено можливість вибрати одну з чотирьох відповідей;

– перевірено можливість обрати тему для раунда методом виключення;

– перевірено перегляд інформації про користувача;

– перевірено перегляд стану паузи;

5) міні-гра Turing Test:

– перевірено можливість змінити команду;

– перевірено можливість вписати питання для штучного інтелекту якщо гравець у команді студентів;

– перевірено можливість вписати відповідь на питання якщо гравець у команді професорів;

– перевірено можливість проголосувати за найпідозрілішу, на думку гравця, відповідь;

– перевірено відображення результату гри;

– перевірено відображення рахунку.

У процесі тестування було охоплено функції контролера на стороні клієнта в ігровому програмному додатку для кваліфікаційного проекту, включаючи логіку всіх трьох ігор. Застосування ручного тестування забезпечило детальну увагу до кожної деталі та дозволило виявити суттєві проблеми. Цей метод також виявився гнучким, дозволяючи використовувати нестандартні та інноваційні підходи. Під час тестування було проведено перевірку різних аспектів функціональності, таких як баланс гри, можливість взаємодії гравців через лобі, можливість паузи гри, а також перевірено кожну з трьох міні-ігор окремо. Кожен аспект був систематично перевірений, що включало в себе можливості взаємодії, зміни налаштувань та відображення різних елементів гри.

Після завершення тестування були створені тест-кейси для деяких функціональних вимог, які наведені в додатку Д.

6 ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

React – JavaScript-бібліотека з відкритим вихідним кодом для розроблення користувацьких інтерфейсів. Оскільки ігровий додаток розробляється з використанням цієї бібліотеки, його можна розмістити на хмарній платформі. Для цього було обрано Render. Render – це уніфікована хмара для створення та запуску всіх програм і веб-сайтів з безкоштовними TLS-сертифікатами, глобальною CDN, приватними мережами та автоматичним розгортанням з Git.

Перед розгортанням, до файлу package.json було вставлено команду start (див. рис. 6.1). Ця команда призначена для запуску програми у виробничому середовищі.

Рисунок 6.1 – Скрипти для керування різними етапами життєвого циклу у файлі package-json front-end частини (знімок екрана виконано самостійно)

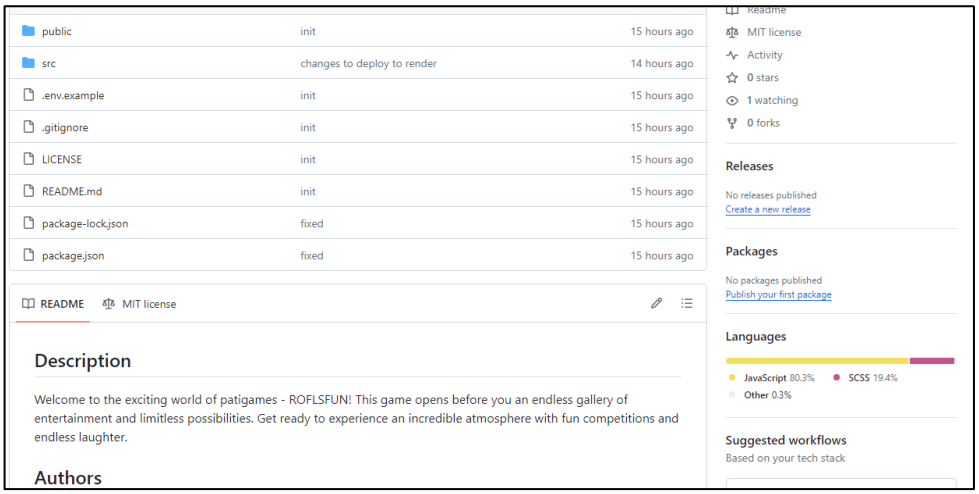


Рисунок 6.2 – GitHub-репозиторій з файлами front-end частини (знімок екрана виконано самостійно)

Потім було створено безкоштовний веб-сервіс, де треба спочатку до платформи Render підключити акаунт на GitHub. Тут задано команди для запуску проєкту (див. рис. 6.3).

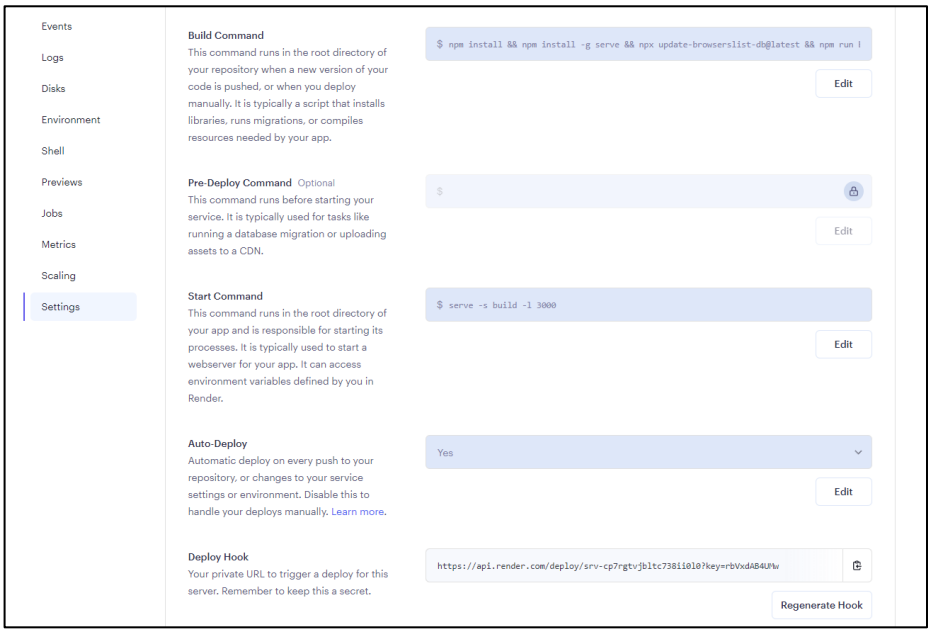


Рисунок 6.3 – Налаштування веб-сервісу на платформі Render (знімок екрана виконано самостійно)

Головним пунктом для коректної роботи – додати змінні середовища з файлу .env (див. рис. 6.4).

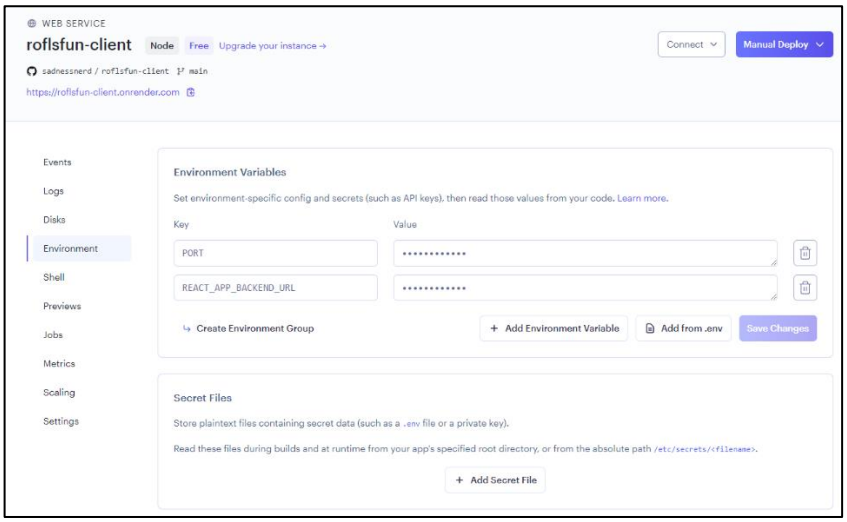


Рисунок 6.4 – Змінні середовища за файлу .env (знімок екрана виконано самостійно)

Після завершення налаштування проєкту на платформі Render, було підтверджено, що ігровий сеанс працює коректно (див. рис. 6.5).

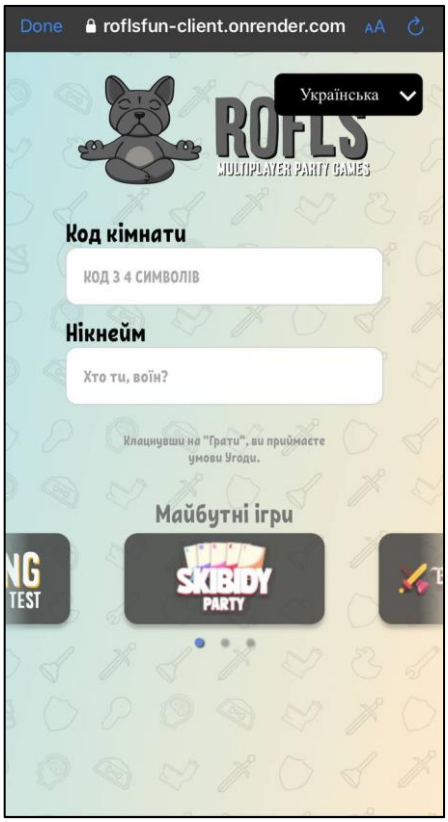


Рисунок 6.5 – Розгорнута ігрова сесія (знімок екрана виконано самостійно)

В результаті, front-end ігрового програмного застосунку була реалізована в Інтернеті.

Відвідування конференції також було одним з етапів впровадження. «Інформаційні інтелектуальні системи» на 28-му Міжнародному форумі «Радіоелектроніка та молодь у XXI столітті», де було представлено тези на тему «Балансування часу щодо складності ігрової ситуації в multiplayer party games, ключові аспекти для створення захопливого ігрового процесу» (див. додаток И). Ця робота була визнана в ході роботи над розділом №3. «Програмна інженерія. Інформаційні технології в освіті» та нагороджено грамотою за змістовну доповідь (див. додаток К).

Іншим етапом впровадження стала участь у виставці технічної творчості молоді під час 28-го Міжнародного форуму «Радіоелектроніка та молодь у XXI столітті» (див. додаток Ж). Команда проекту ROFLSUN отримала друге місце на цій виставці, представивши ігровий додаток у жанрі multiplayer party games. (див. додаток И).

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було розроблено балансування ігрового процесу та мобільний контролер в ігровий програмний застосунок в жанрі multiplayer party games.

Одним з головних завдань ретельно продумати да досягти балансу у грі [12]. Для цього були внесені різні зміни в часові обмеження, механіку гри та складність завдань з метою забезпечення рівних умов для всіх гравців і підтримки зацікавленості протягом усієї гри. Важливим етапом було встановлення адекватного рівня виклику та надання можливостей для стратегічного мислення та швидкого реагування.

Розробка мобільного програмного ігрового контролера була ключовим аспектом проекту, ретельно організованого на клієнтській стороні додатку. Використовуючи можливості мови програмування JavaScript та бібліотеки React, було розроблено адаптивний та інтуїтивно зрозумілий інтерфейс. Інтеграція бібліотеки Socket.IO, полегшила обмін даними в режимі реального часу між клієнтськими та серверними компонентами, забезпечуючи плавний ігровий процес та безперебійне з’єднання. В рамках роботи цієї структури, було інтегровано набір функцій у модуль мобільного контролера. Це включає в себе керування грою, створенням і управління гральними кімнатами, обробку дій гравців та вивід даних про гру та користувача. Ця частина застосунку забезпечує ефективне керування ігровим процесом. Підхід керуванню через смартфон 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 успішній реалізації цілей з балансу та інтеграції технології мобільних контролерів, створено приємний та захоплюючий ігровий досвід.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Harteveld C. Triadic Game Design: Balancing Reality, Meaning and Play. Springer London, Limited, 2011.

2. Koss H. These design principles made jackbox a party game phenomenon | built in. *Built In*. URL: https://builtin.com/articles/jackbox-games-design-party-pack (дата звернення: 25.05.2024).

3. Juul J. Fear of Failing? The Many Meanings of Difficulty in Video Games. *Jesper Juul*. URL: https://www.jesperjuul.net/text/fearoffailing/ (дата звернення: 25.05.2024).

4. Мірошніков Є. В. Балансування часу щодо складності ігрової ситуації в multiplayer party games, ключові аспекти для створення захопливого ігрового процесу / Є. В. Мірошніков // Радіоелектроніка та молодь у XXI столітті: тези доповідей 28-го Міжнародного молодіжного форуму, 16–18 квітня 2024 р. – Харків : ХНУРЕ, 2024. – Т. 6. – С. 372–374.

5. Adams E., Dormans J. Game mechanics: advanced game design. New Riders Publishing, 2012. 360 с.

6. Nixon R. JavaScript: 20 lessons to successful web development. McGraw-Hill Education, 2015. 256 с.

7. Getting started – react. *React – A JavaScript library for building user interfaces*. URL: https://legacy.reactjs.org/docs/getting-started.html (дата звернення: 25.05.2024).

8. Jalloul G. UML by Example. Cambridge University Press, 2012.

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

10. 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).

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

12. Давидов О. П. Оптимізація ігрового балансування стратегій з використанням математичних алгоритмів теорії ігор / О. П. Давидов // Радіоелектроніка та молодь у XXI столітті: тези доповідей 22-го Міжнародного молодіжного форуму, 2019 р. – Харків : ХНУРЕ, 2019. – Т. 6. – С. 496.

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

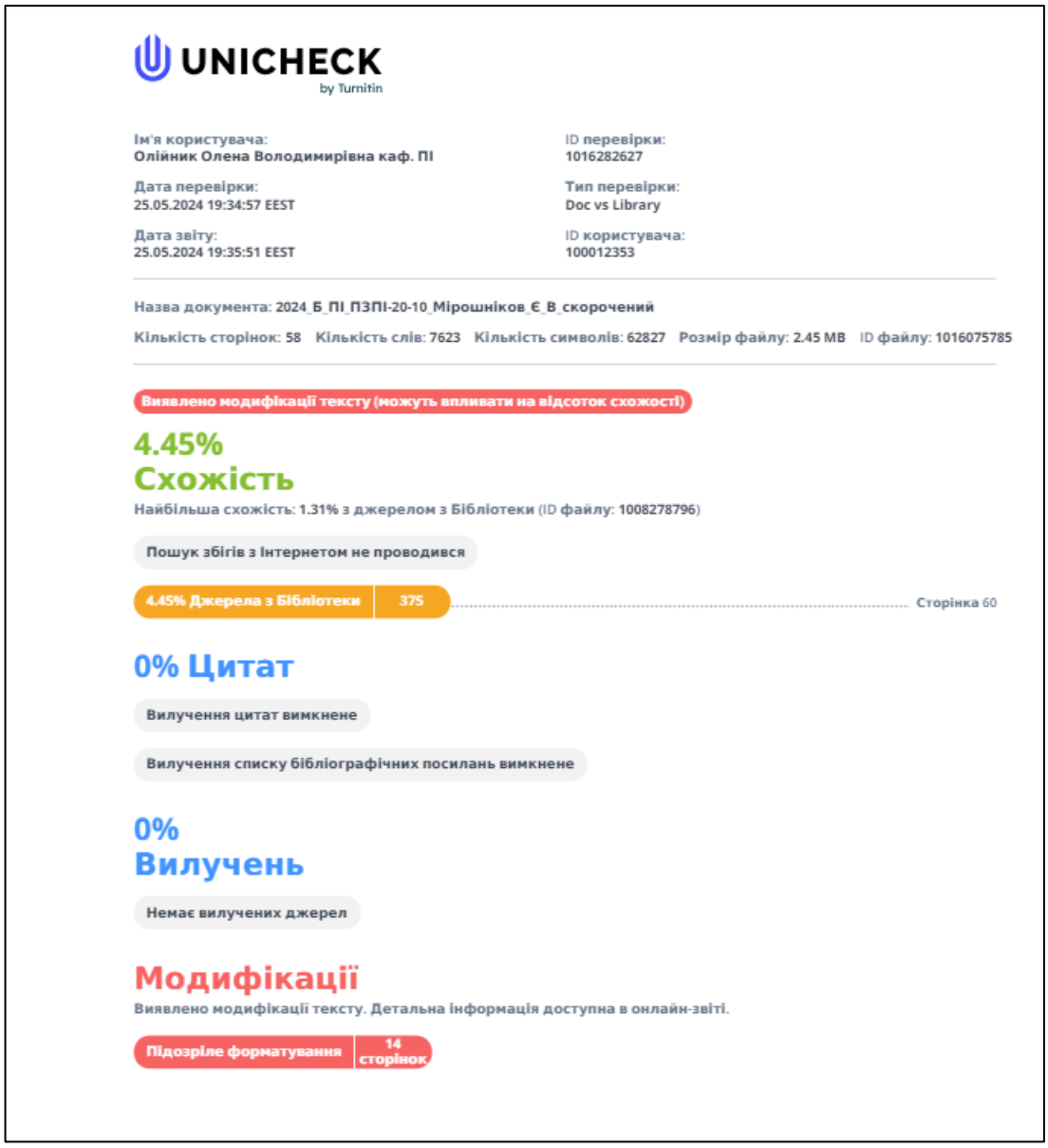


Рисунок А.1 – Результат перевірки на унікальність тексту в базі ХНУРЕ (знміок екрана виконано самостійно)

ДОДАТОК Б

Слайди презентації

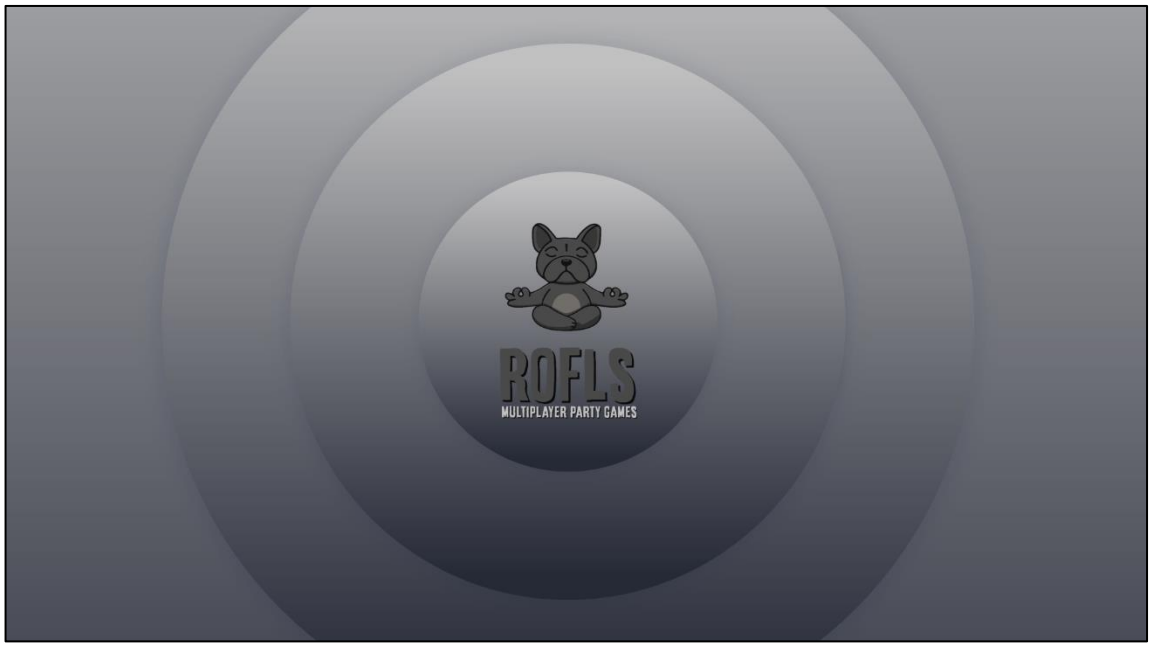


Рисунок Б.1 – Перший слайд презентації (знімок екрана виконано самостійно)

Рисунок Б.2 – Другий слайд презентації (знімок екрана виконано самостійно)

Рисунок Б.3 – Третій слайд презентації (знімок екрана виконано самостійно)

Рисунок Б.4 – 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)

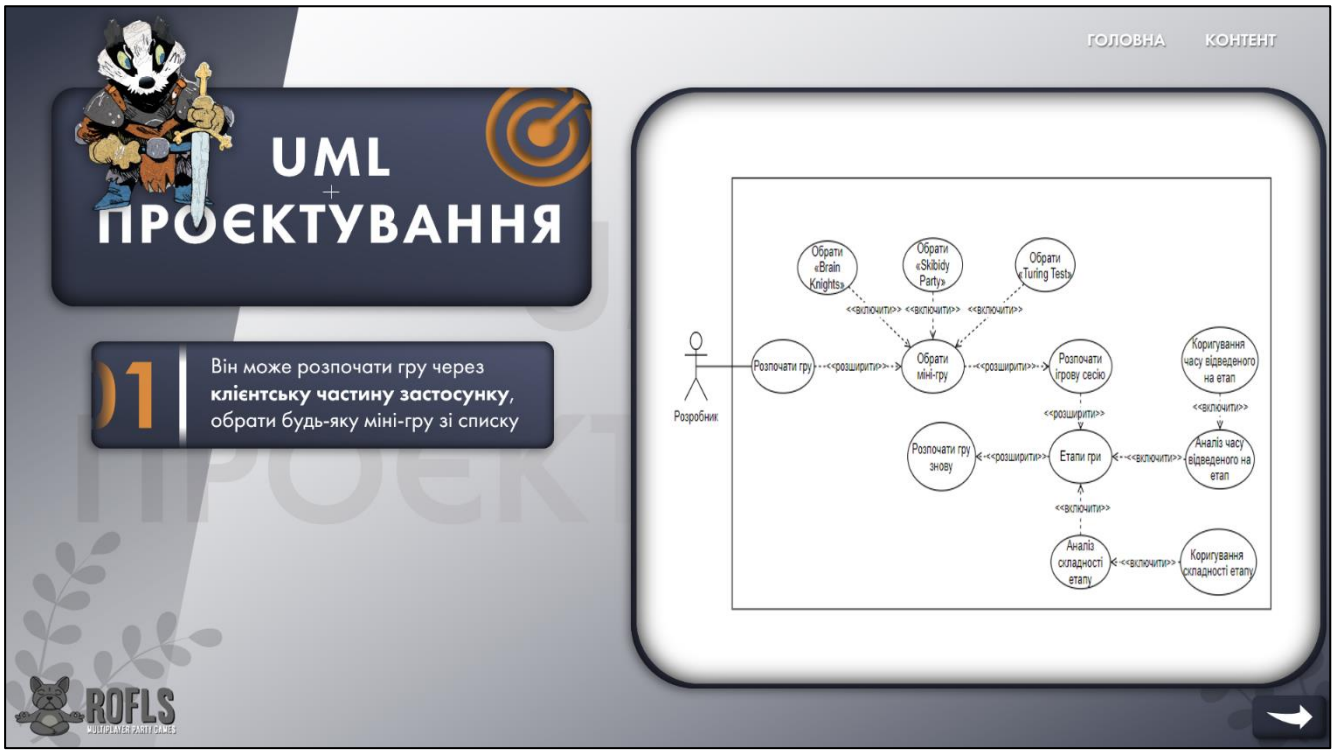


Рисунок Б.9 – Дев’ятий слайд презентації (знімок екрана виконано самостійно)

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

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

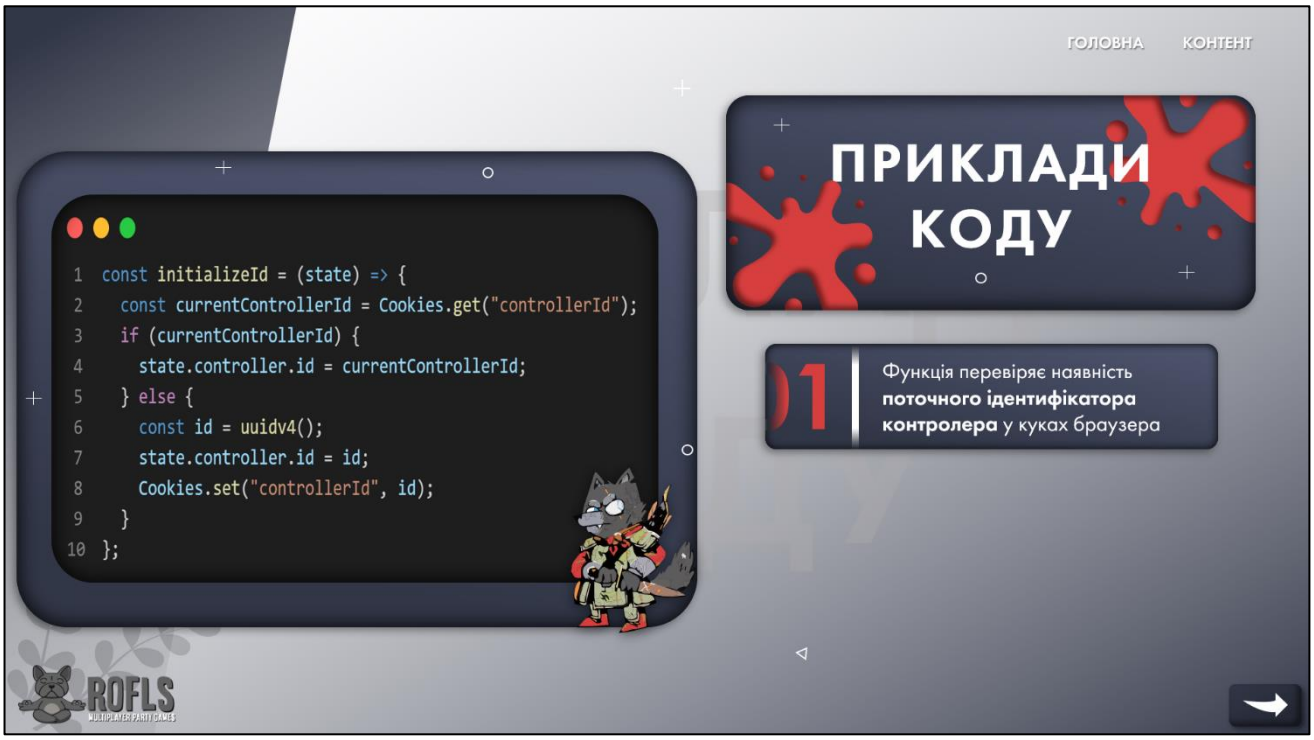


Рисунок Б.12 – Дванадцятий слайд презентації (знімок екрана виконано самостійно)

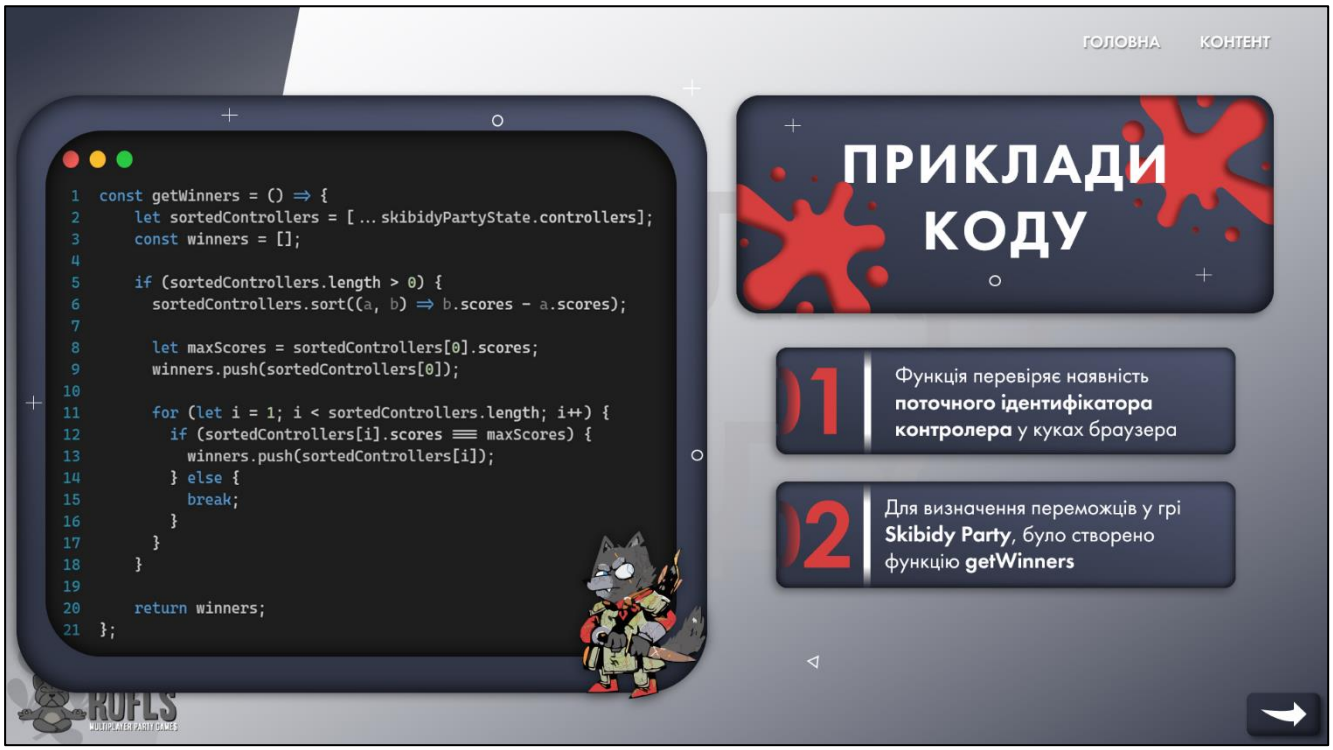


Рисунок Б.13 – Тринадцятий слайд презентації (знімок екрана виконано самостійно)

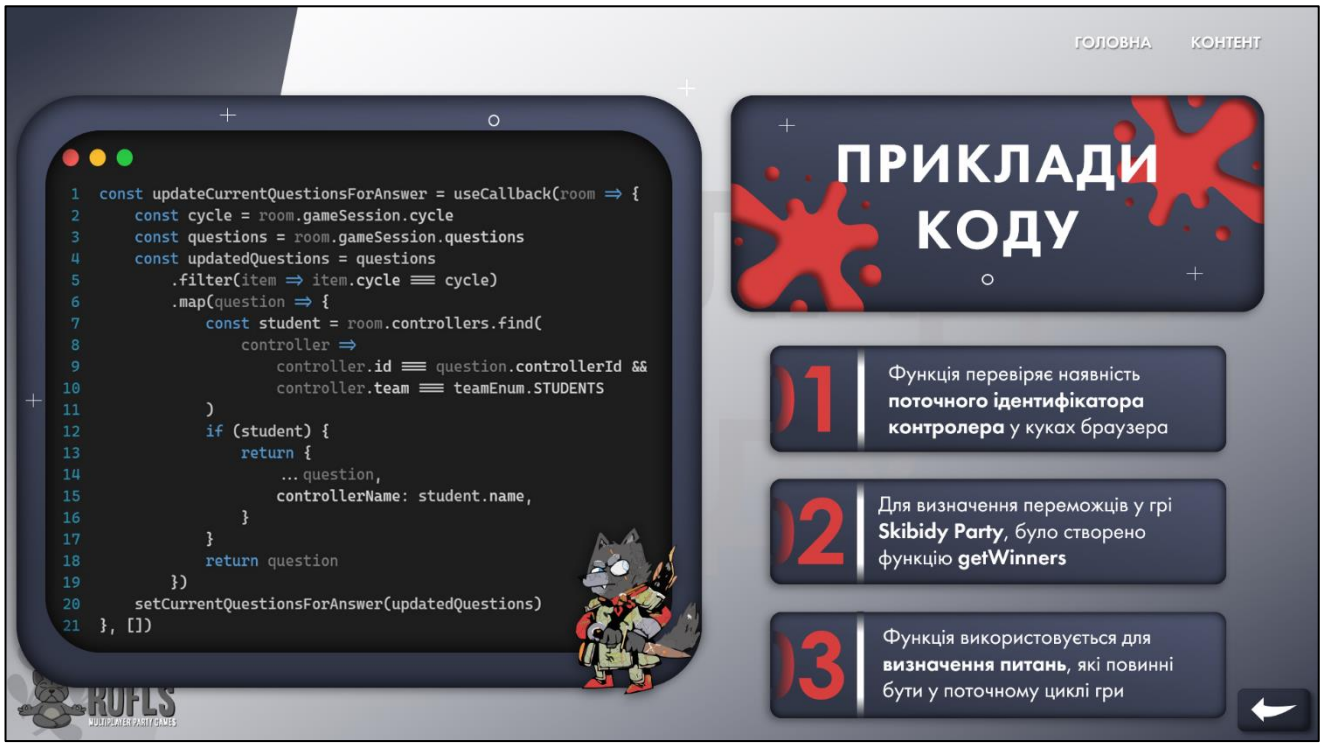


Рисунок Б.14 – Чотирнадцятий слайд презентації (знімок екрана виконано самостійно)

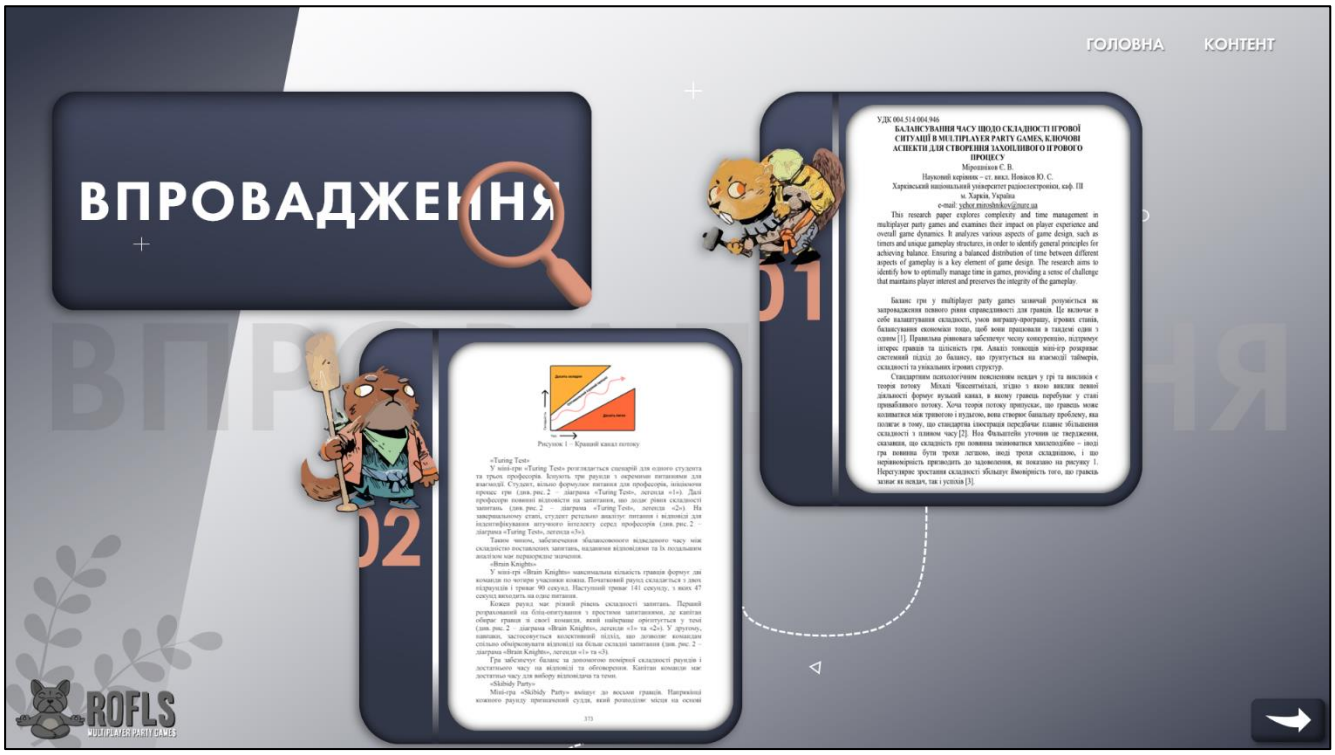


Рисунок Б.15 – П’ятнадцятий слайд презентації (знімок екрана виконано самостійно)

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

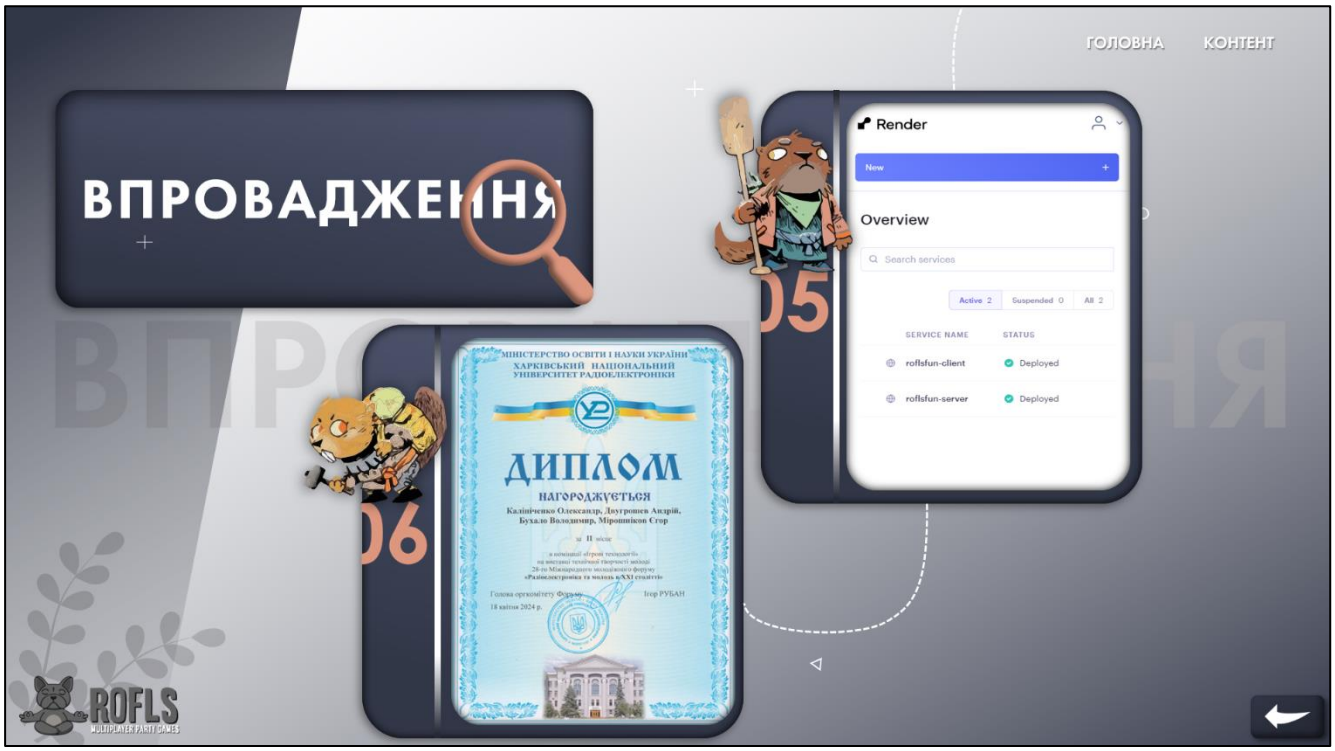


Рисунок Б.17 – Сімнадцятий слайд презентації (знімок екрана виконано самостійно)

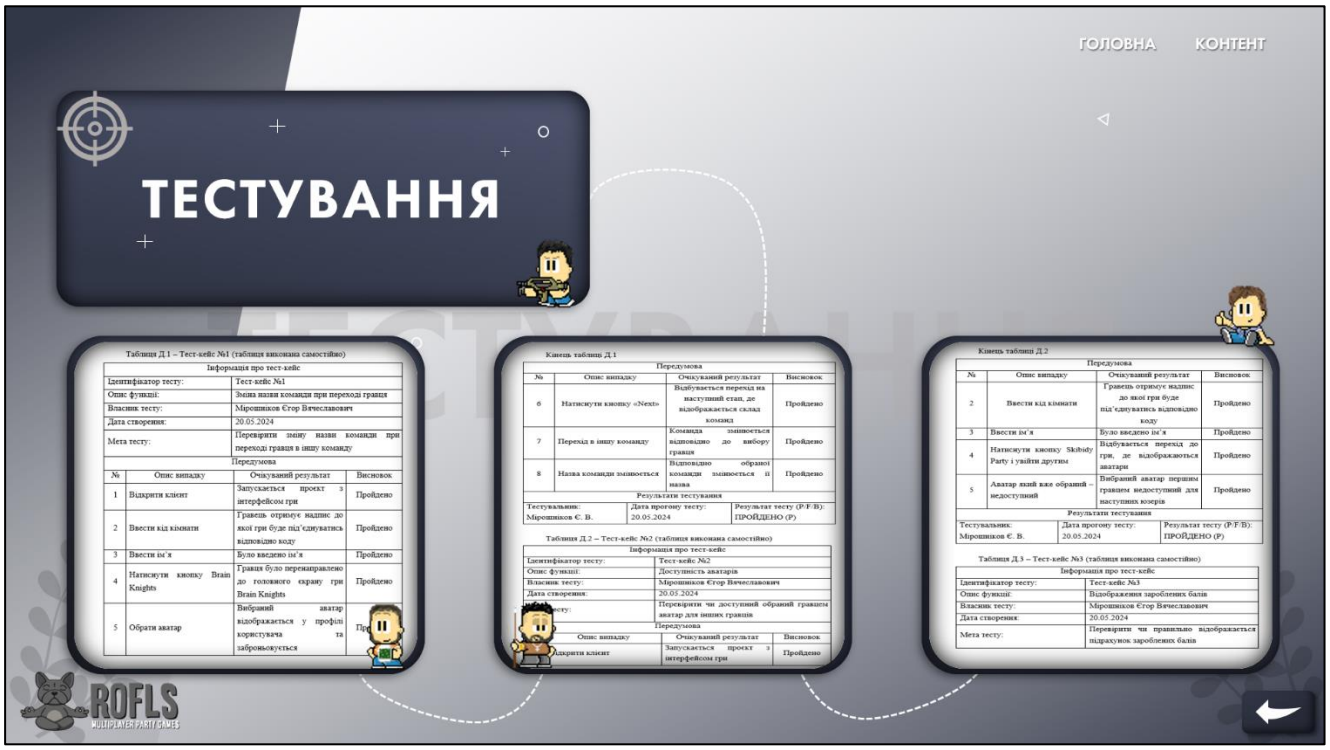


Рисунок Б.18 – Вісімнадцятий слайд презентації (знімок екрана виконано самостійно)

Рисунок Б.19 – Дев’ятнадцятий слайд презентації (знімок екрана виконано самостійно)

Рисунок Б.20 – Двадцятий 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.

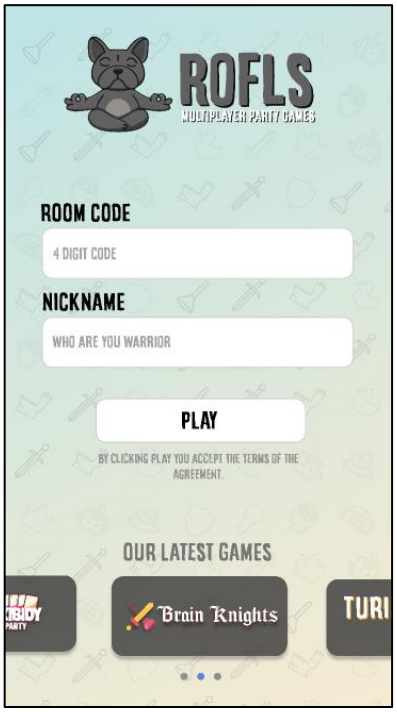


Рисунок 3.2 – Інтерфейс стартового екрану контролера (знімок екрана виконано самостійно)

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.

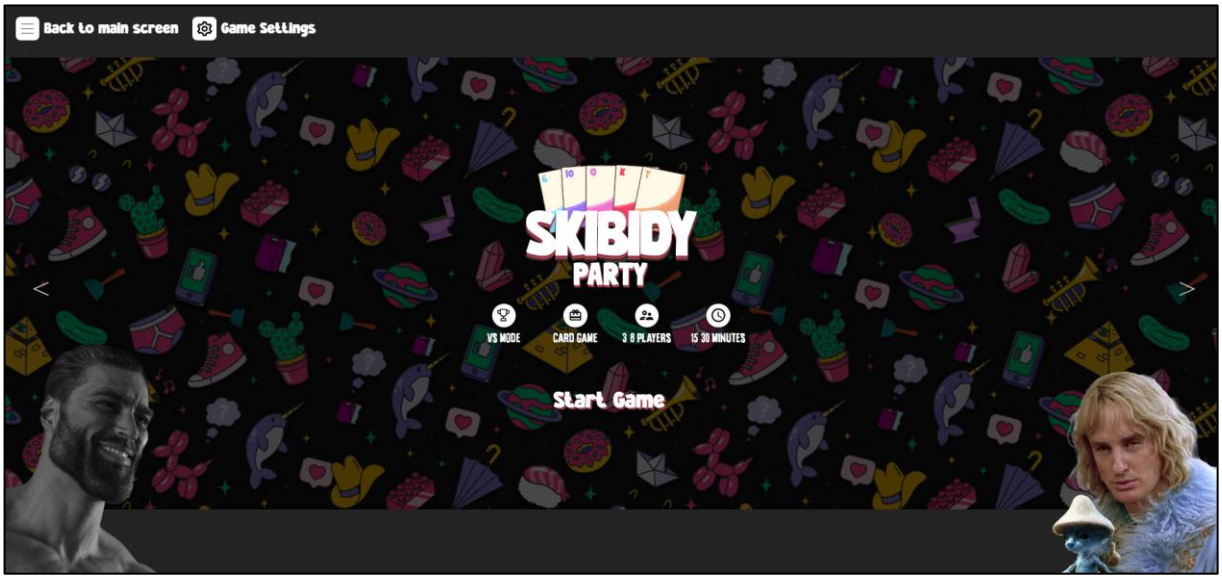


Рисунок 3.4 – Вибір гри Skibidy Party (знімок екрана виконано самостійно)

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

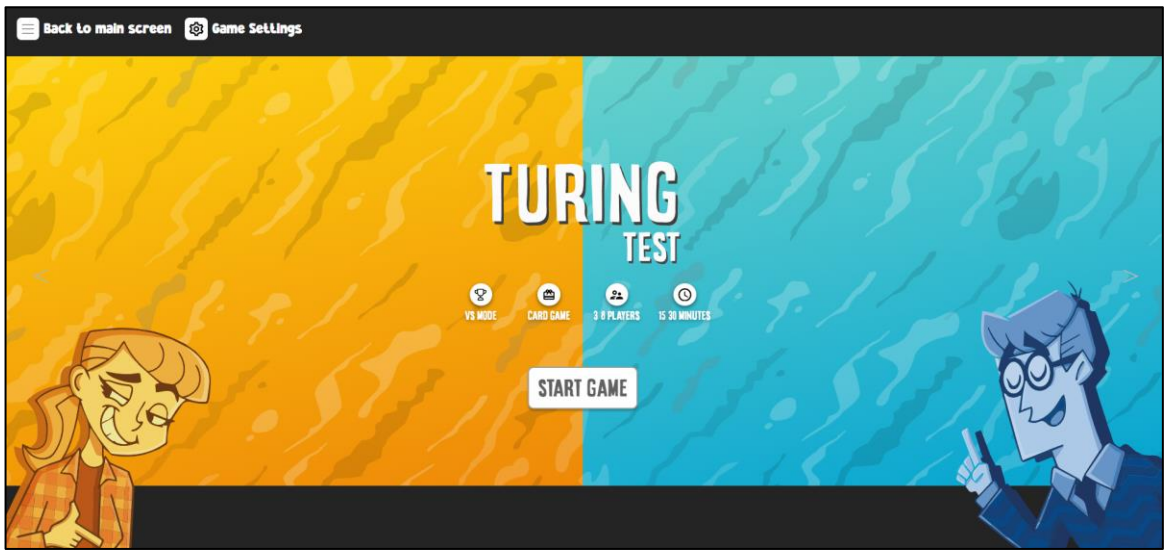


Рисунок 3.5 – Вибір гри Turing Test (знімок екрана виконано самостійно)

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

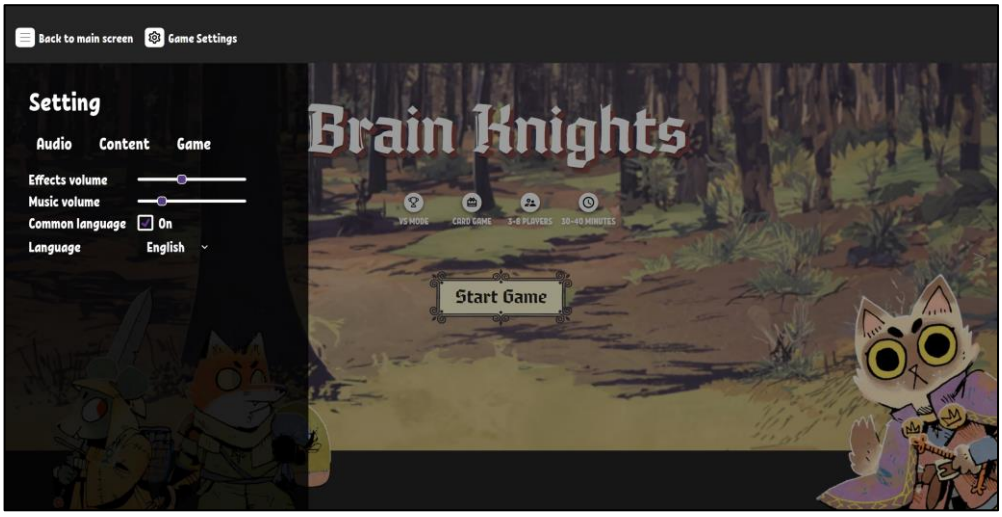


Рисунок 3.6 – Налаштуванні ігрового процесу (знімок екрана виконано самостійно)

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.

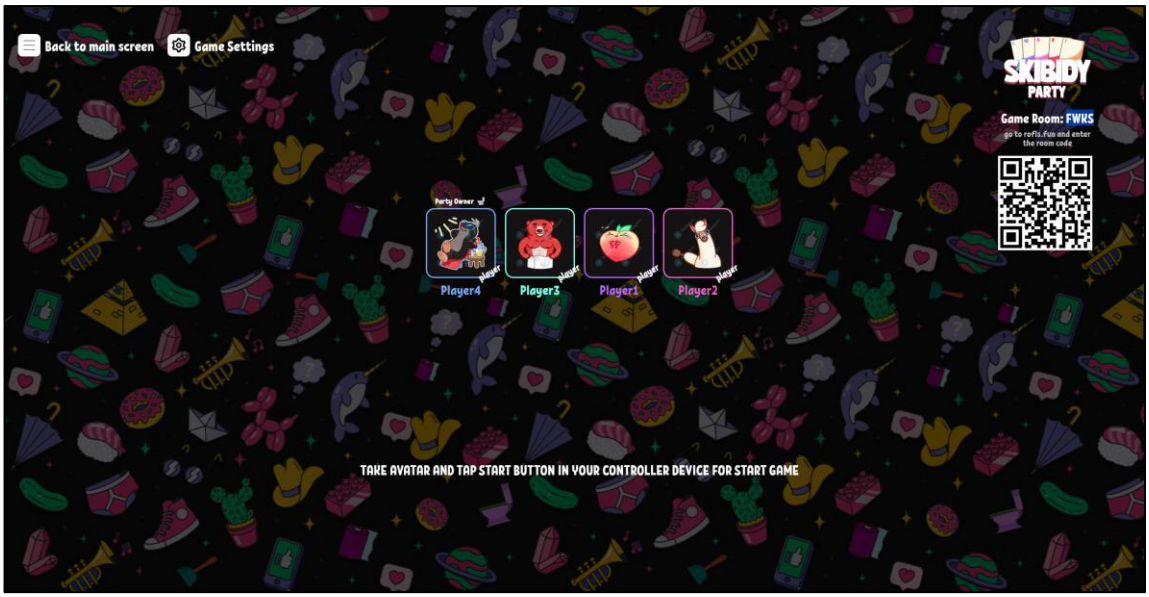


Рисунок 3.7 – Інтерфейс ігрової кімнати в грі Skibidy Party на хості (знімок екрана виконано самостійно)

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

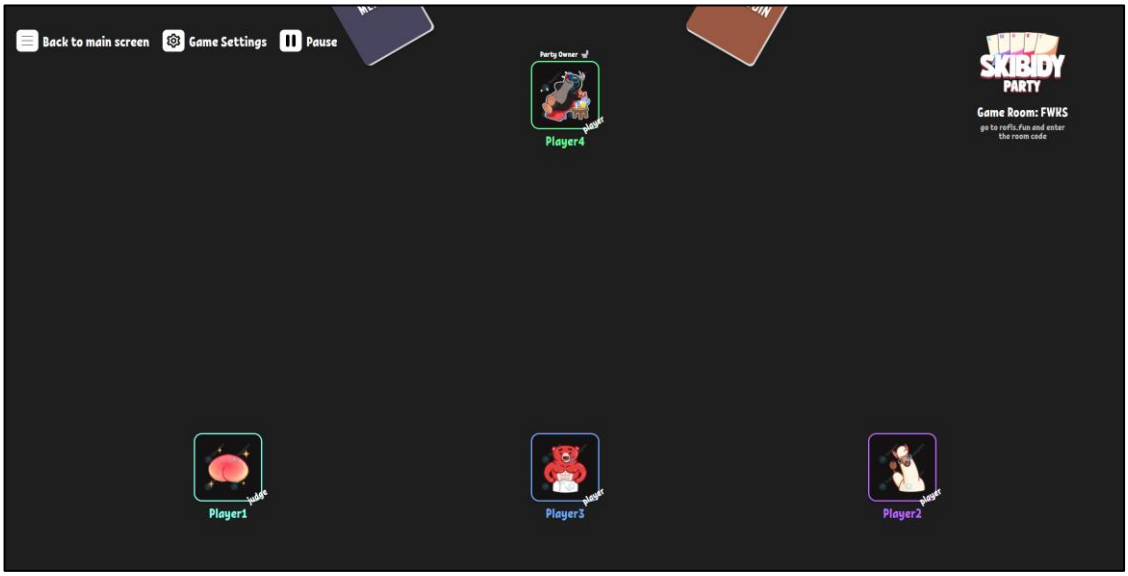


Рисунок 3.8 – Інтерфейс ігрового поля в Skibidy Party на хості (знімок екрана виконано самостійно)

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

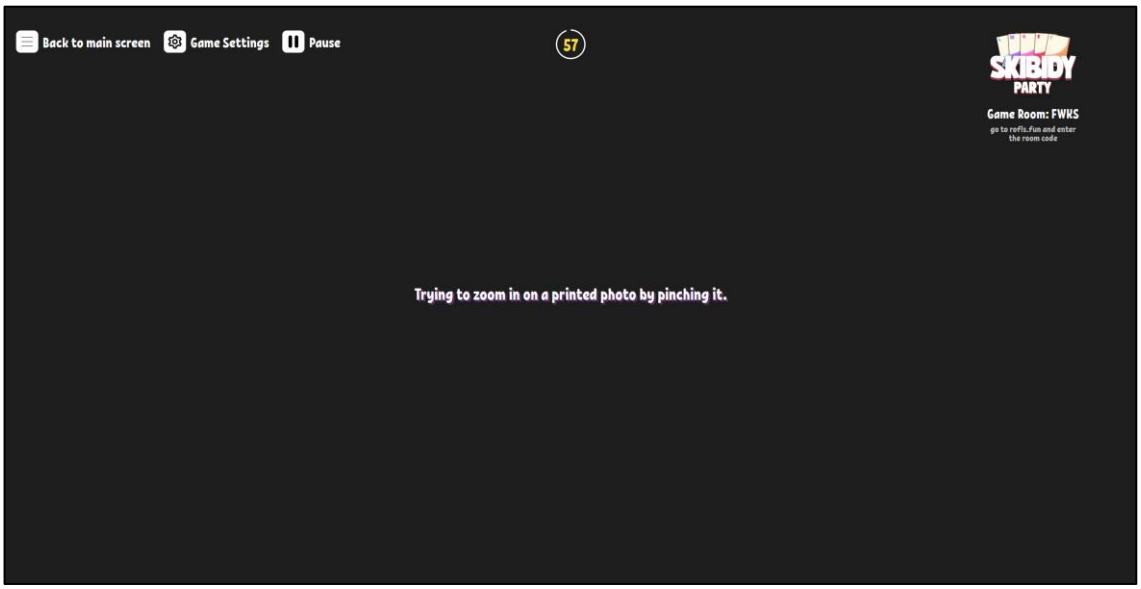


Рисунок 3.9 – Інтерфейс ігрової ситуації в Skibidy Party на хості (знімок екрана виконано самостійно)

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

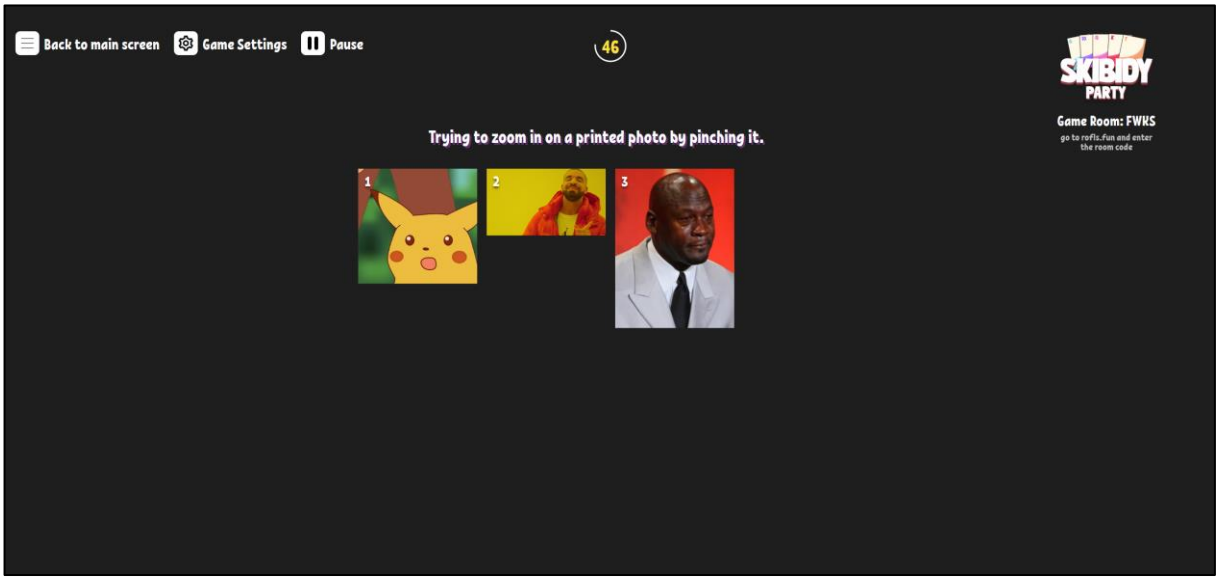


Рисунок 3.10 – Інтерфейс обраних мемів в Skibidy Party на хості (знімок екрана виконано самостійно)

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

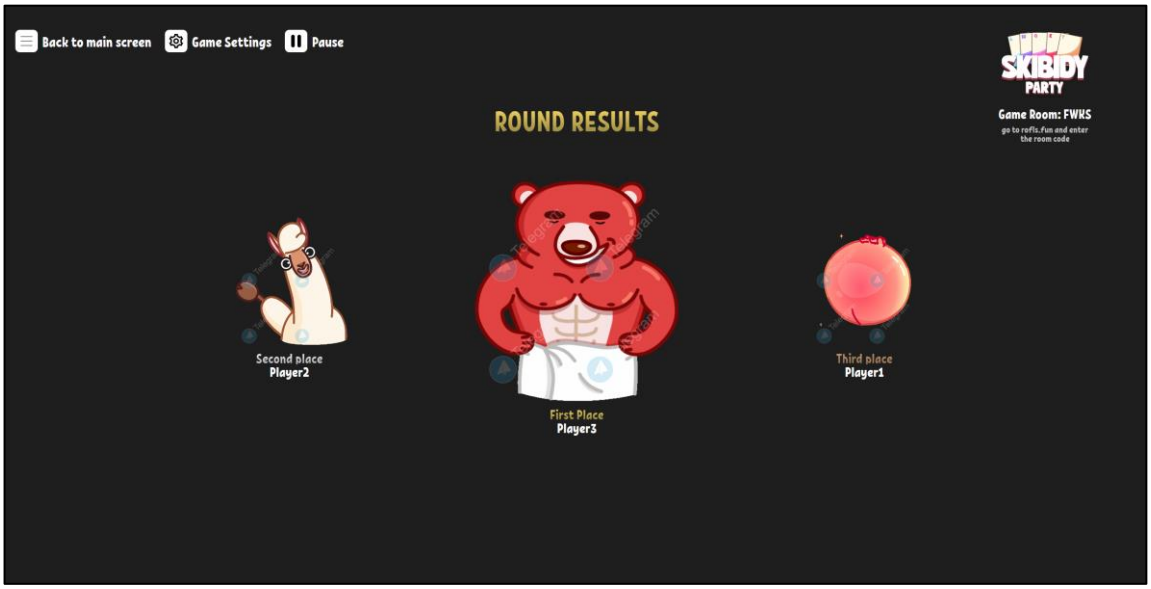


Рисунок 3.11 – Інтерфейс переможців раунду в Skibidy Party на хості (знімок екрана виконано самостійно)

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.

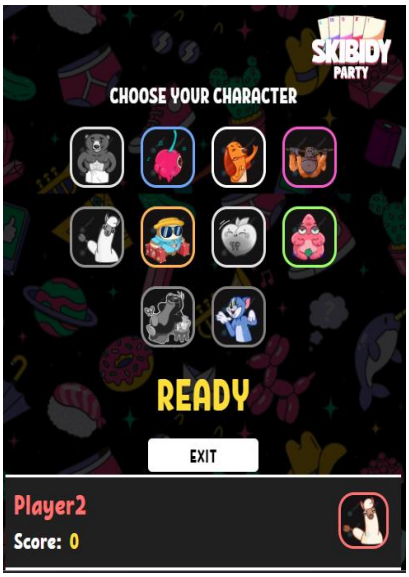


Рисунок 3.13 – Вибір аватару в Skibidy Party на контролері (знімок екрана виконано самостійно)

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

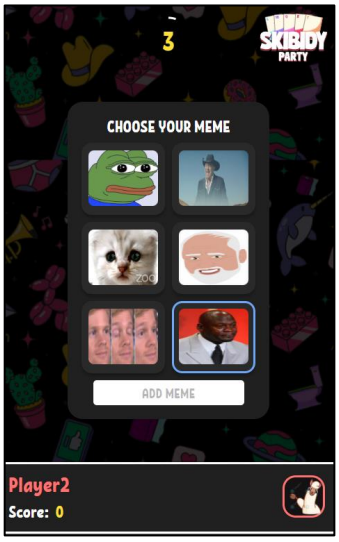


Рисунок 3.14 – Вибір мему в Skibidy Party на контролері (знімок екрана виконано самостійно)

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

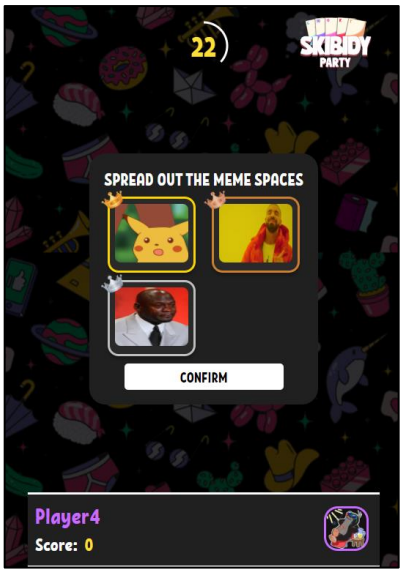


Рисунок 3.15 – Вибір переможного мему в Skibidy Party на контролері (знімок екрана виконано самостійно)

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.

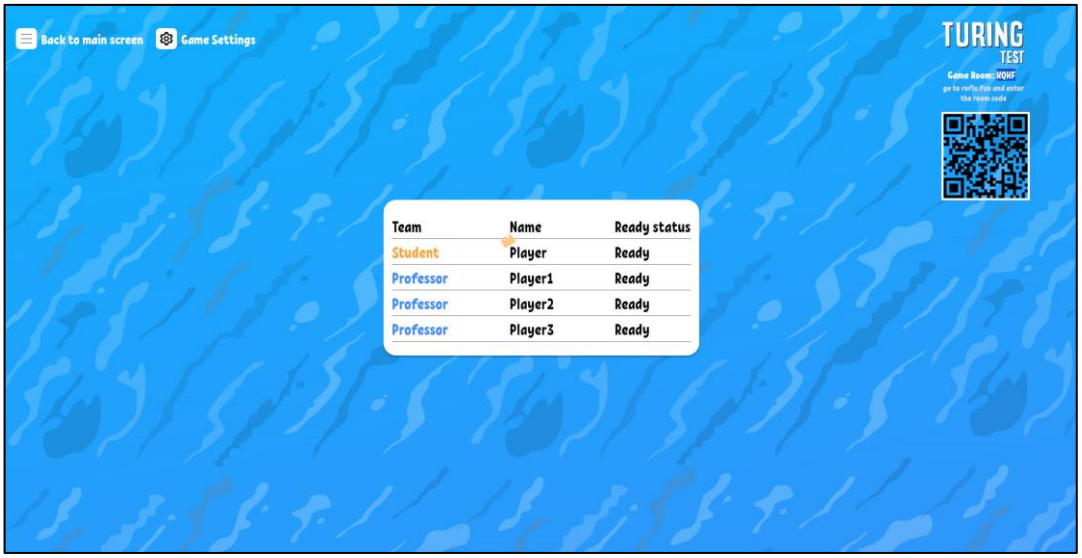


Рисунок 3.16 – Інтерфейс ігрової кімнати в грі Turing Test на хості (знімок екрана виконано самостійно)

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.

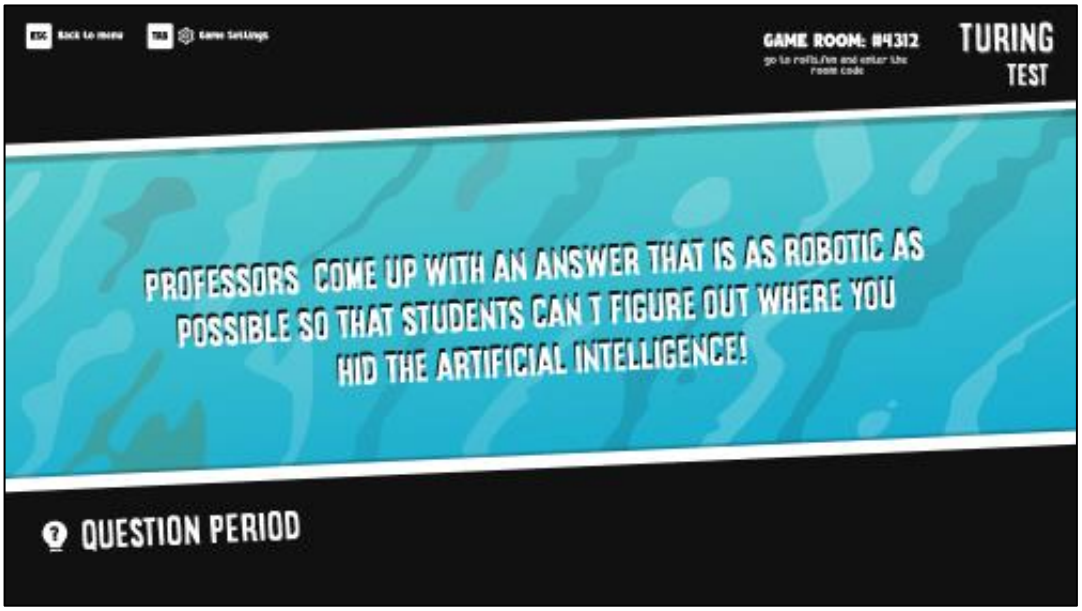


Рисунок 3.18 – Інтерфейс раунду в грі Turing Test на хості (знімок екрана виконано самостійно)

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.

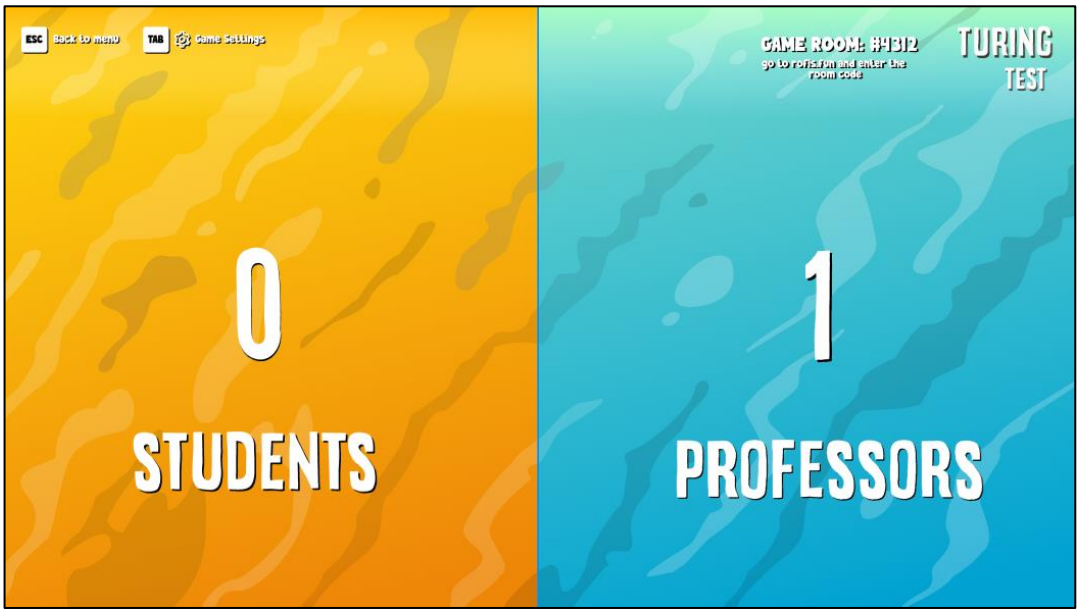


Рисунок 3.23 – Інтерфейс рахунку в грі Turing Test на хості (знімок екрана виконано самостійно)

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.

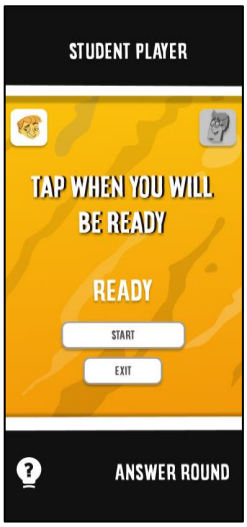


Рисунок 3.24 – Інтерфейс вибору команди в грі Turing Test на котролері (знімок екрана виконано самостійно)

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».

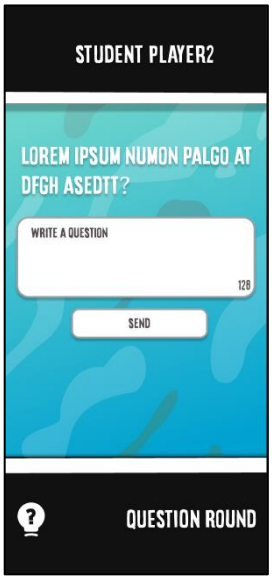


Рисунок 3.26 – Інтерфейс написання відповіді на питання професорами в грі Turing Test на котролері (знімок екрана виконано самостійно)

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.

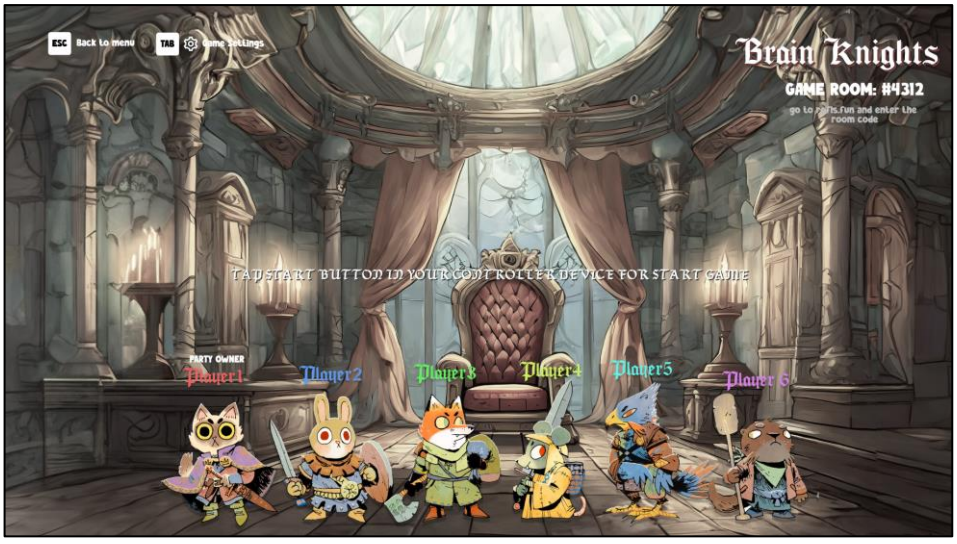


Рисунок 3.29 – Інтерфейс ігрової кімнати на головному екрані гри в грі Brain Knights на хості (знімок екрана виконано самостійно)

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

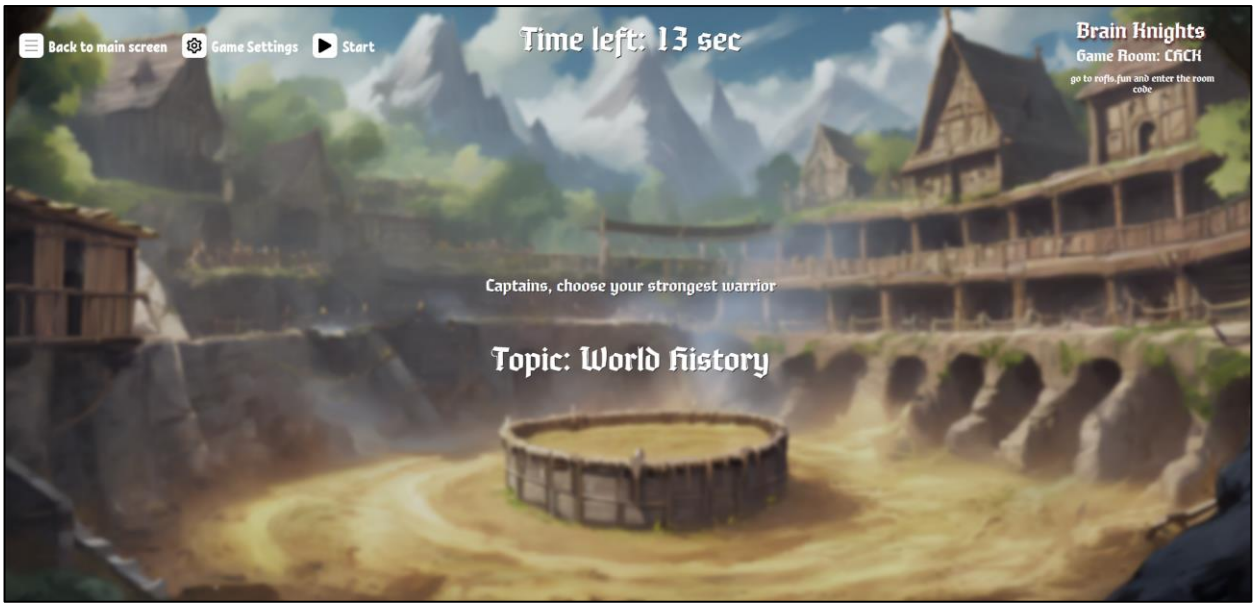


Рисунок 3.30 – Інтерфейс теми раунду в грі Brain Knights на хості (знімок екрана виконано самостійно)

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

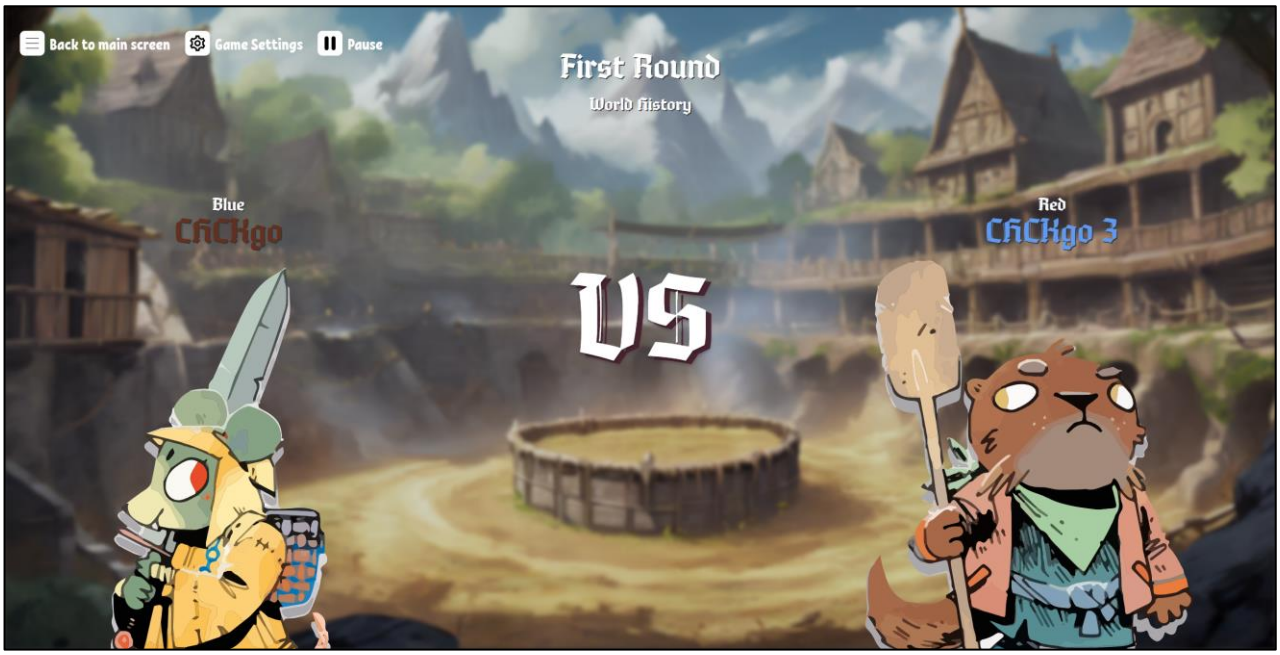


Рисунок 3.31 – Інтерфейс початку бліц-раунду в грі Brain Knights на хості (знімок екрана виконано самостійно)

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

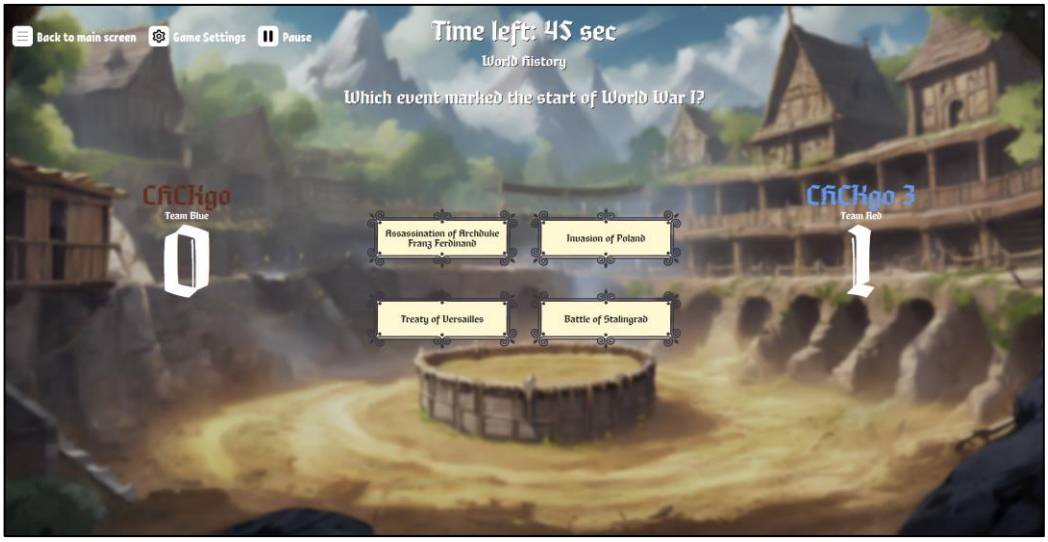


Рисунок 3.32 – Інтерфейс ігрового раунду в грі Brain Knights на хості (знімок екрана виконано самостійно)

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.

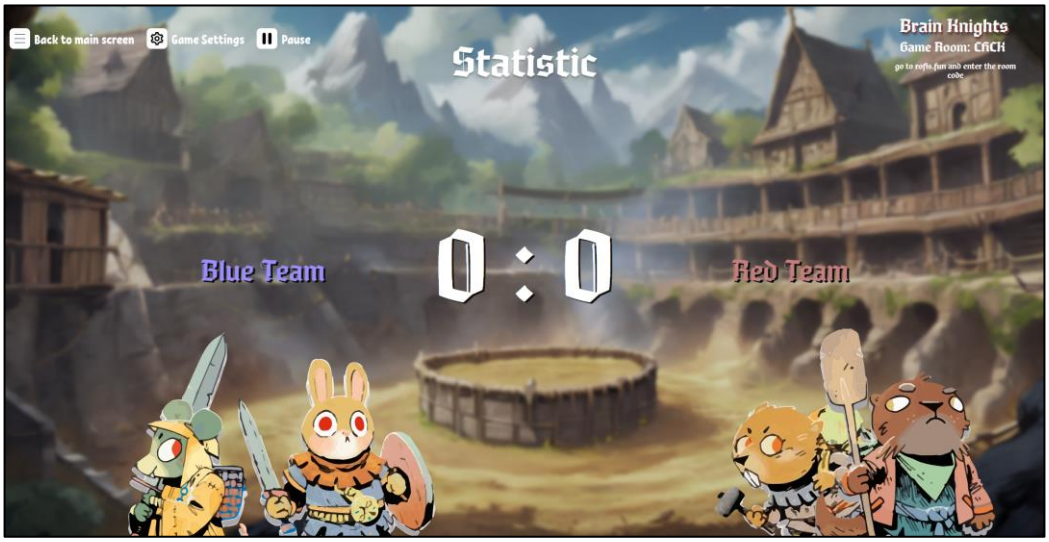


Рисунок 3.34 – Проміжні результати раунду в грі Brain Knights на хості (знімок екрана виконано самостійно)

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.

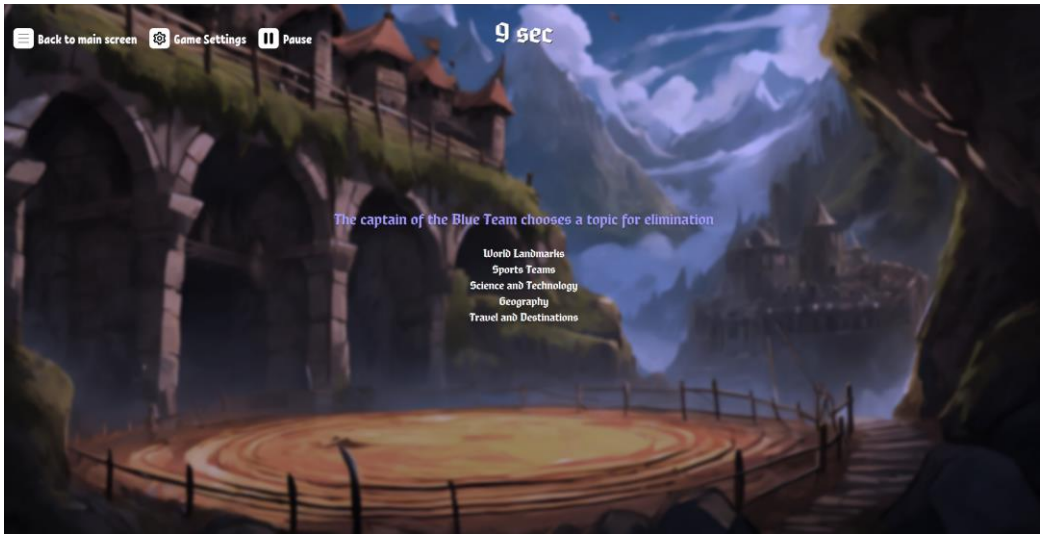


Рисунок 3.35 – Вибір теми для другої стадії в грі Brain Knights на хості (знімок екрана виконано самостійно)

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

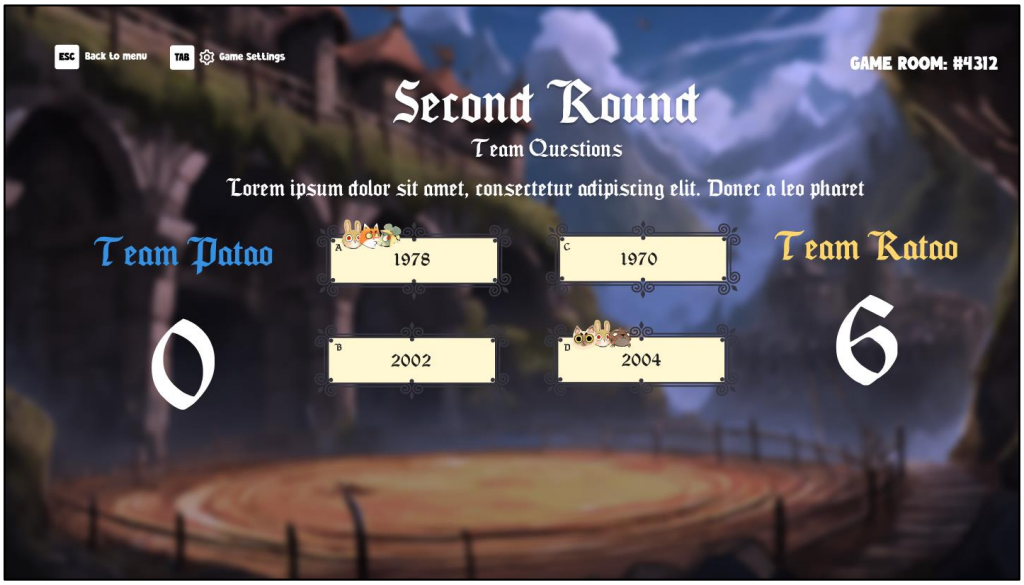


Рисунок 3.36 – Ігровий раунд у другій стадії в грі Brain Knights на хості (знімок екрана виконано самостійно)

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.

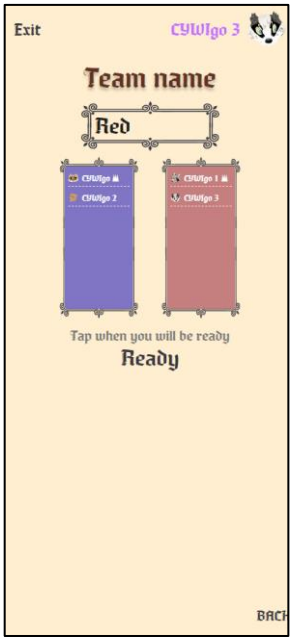


Рисунок 3.38 – Вибір команди в грі Brain Knights на хості (знімок екрана виконано самостійно)

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

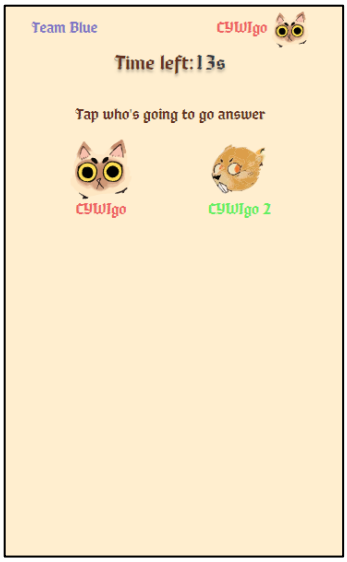


Рисунок 3.39 – Вибір гравця для відповіді в грі Brain Knights на хості (знімок екрана виконано самостійно)

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

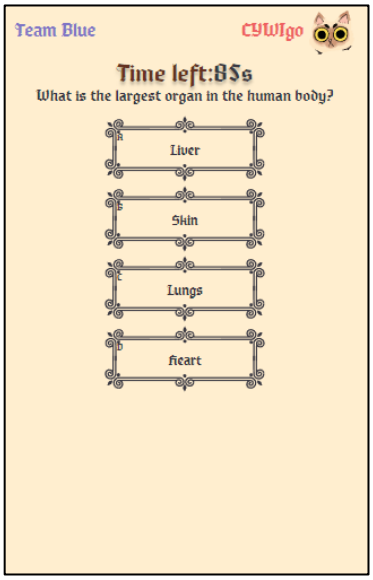


Рисунок 3.40 – Вибір відповіді на запитання в грі Brain Knights на хості (знімок екрана виконано самостійно)

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

Додаткових вимог нем

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 back-end частини ігрового програмного застосунку у жанрі multiplayer party games

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

Інформація про тест-кейс			
Ідентифікатор тесту:	Тест-кейс №1		
Опис функції:	Зміна назви команди при переході гравця		
Власник тесту:	Мірошніков Єгор Вячеславович		
Дата створення:	20.05.2024		
Мета тесту:	Перевірити зміну назви команди при переході гравця в іншу команду		
Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	Відкрити клієнт	Запускається проєкт з інтерфейсом гри	Пройдено
2	Ввести кід кімнати	Гравець отримує надпис до якої гри буде під’єднуватись відповідно коду	Пройдено
3	Ввести ім’я	Було введено ім’я	Пройдено
4	Натиснути кнопку Brain Knights	Гравця було перенаправлено до головного єкрану гри Brain Knights	Пройдено
5	Обрати аватар	Вибраний аватар відображається у профілі користувача та забронюється	Пройдено

Кінець таблиці Д.1

Передумова			
№	Опис випадку	Очікуваний результат	Висновок
6	Натиснути кнопку «Next»	Відбувається перехід на наступний етап, де відображається склад команд	Пройдено
7	Перехід в іншу команду	Команда змінюється відповідно до вибору гравця	Пройдено
8	Назва команди змінюється	Відповідно обраної команди змінюється її назва	Пройдено
Результати тестування			
Тестувальник: Мірошніков Є. В.	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:	Доступність аватарів		
Власник тесту:	Мірошніков Єгор Вячеславович		
Дата створення:	20.05.2024		
Мета тесту:	Перевірити чи доступний обраний гравцем аватар для інших гравців		
Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	Відкрити клієнт	Запускається проєкт з інтерфейсом гри	Пройдено

Кінець таблиці Д.2

Передумова			
№	Опис випадку	Очікуваний результат	Висновок
2	Ввести кід кімнати	Гравець отримує надпис до якої гри буде під’єднуватись відповідно коду	Пройдено
3	Ввести ім’я	Було введено ім’я	Пройдено
4	Натиснути кнопку Skibidy Party і увійти другим	Відбувається перехід до гри, де відображаються аватари	Пройдено
5	Аватар який вже обраний – недоступний	Вибраний аватар першим гравцем недоступний для наступних юзерів	Пройдено
Результати тестування			
Тестувальник: Мірошніков Є. В.	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:	Відображення зароблених балів
Власник тесту:	Мірошніков Єгор Вячеславович
Дата створення:	20.05.2024
Мета тесту:	Перевірити чи правильно відображається підрахунок зароблених балів

Продовження таблиці Д.3

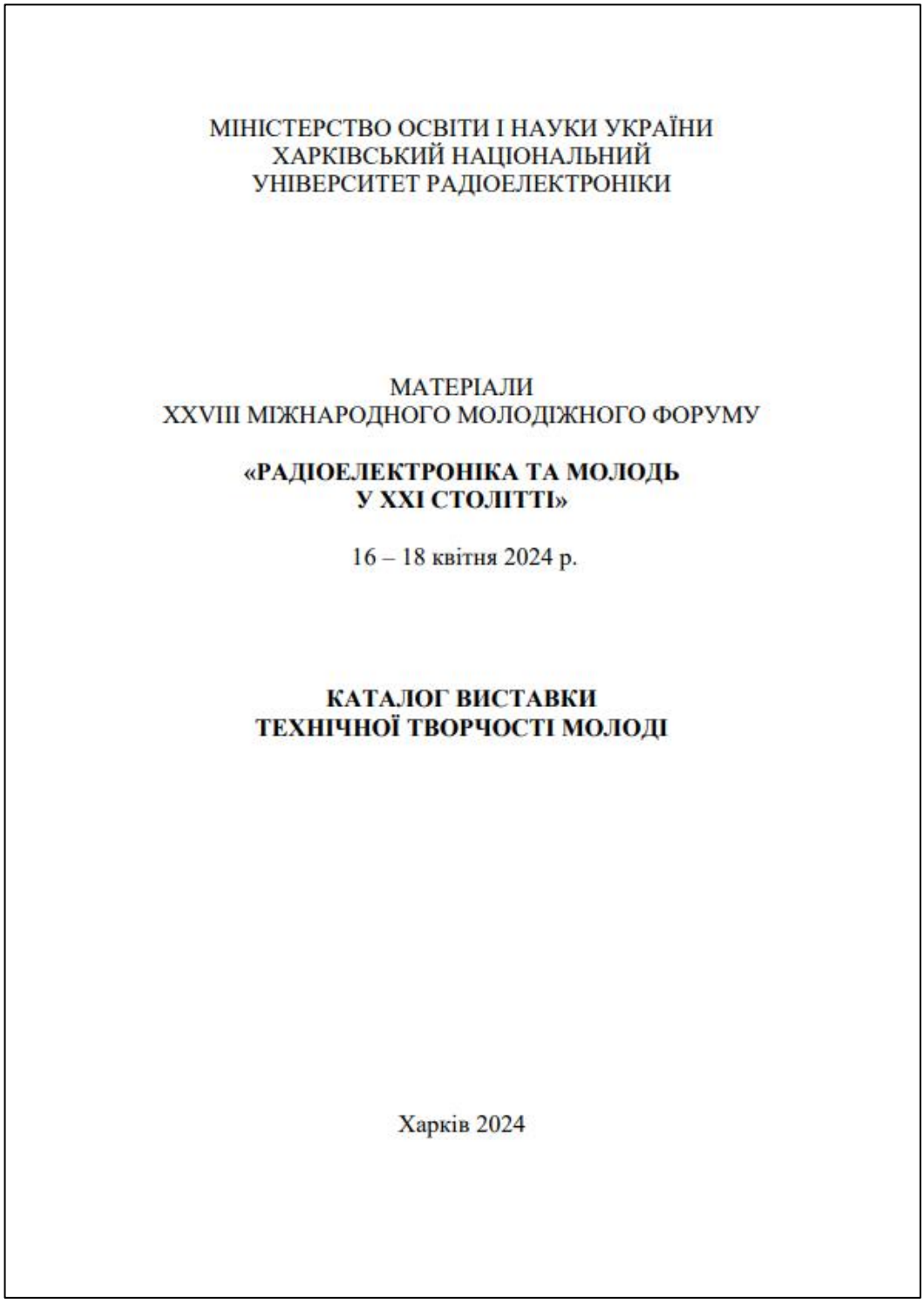
Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	Відкрити клієнт	Запускається проєкт з інтерфейсом гри	Пройдено
2	Ввести кід кімнати	Гравець отримує надпис до якої гри буде під’єднуватись відповідно коду	Пройдено
3	Ввести ім’я	Було введено ім’я	Пройдено
4	Натиснути кнопку Turing Test і увійти другим	Відбувається перехід до головного єкрану гри	Пройдено
5	Обрати команду	Гравець стає професором або студентом	Пройдено
6	Натиснути кнопку «Ready»	Статус гравця переходить до «готовий до початку»	Пройдено
7	Дочекатися інших гравців	Юзер очікує, доки до гри приєднаються інші гравці, і після цього гра розпочинається	Пройдено
8	Натиснути кнопку «Start»	Відбувається перехід до етапів гри та ігровий процес активується	Пройдено
9	Дочекатися етапу, коли повинен з’явитися екран з підрахованими балами	Відображення підрахованих балів системою впродовж гри	Пройдено
10	Бали пораховані належним чином	Бали пораховані точно та відповідають логіці гри	Пройдено

Кінець таблиці Д.3

Результати тестування		
Тестувальник: Мірошніков Є. В.	Дата прогону тесту: 20.05.2024	Результат тесту (P/F/B): ПРОЙДЕНО (P)

ДОДАТОК Е

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 Е.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 Е.2 – Каталог виставки (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»

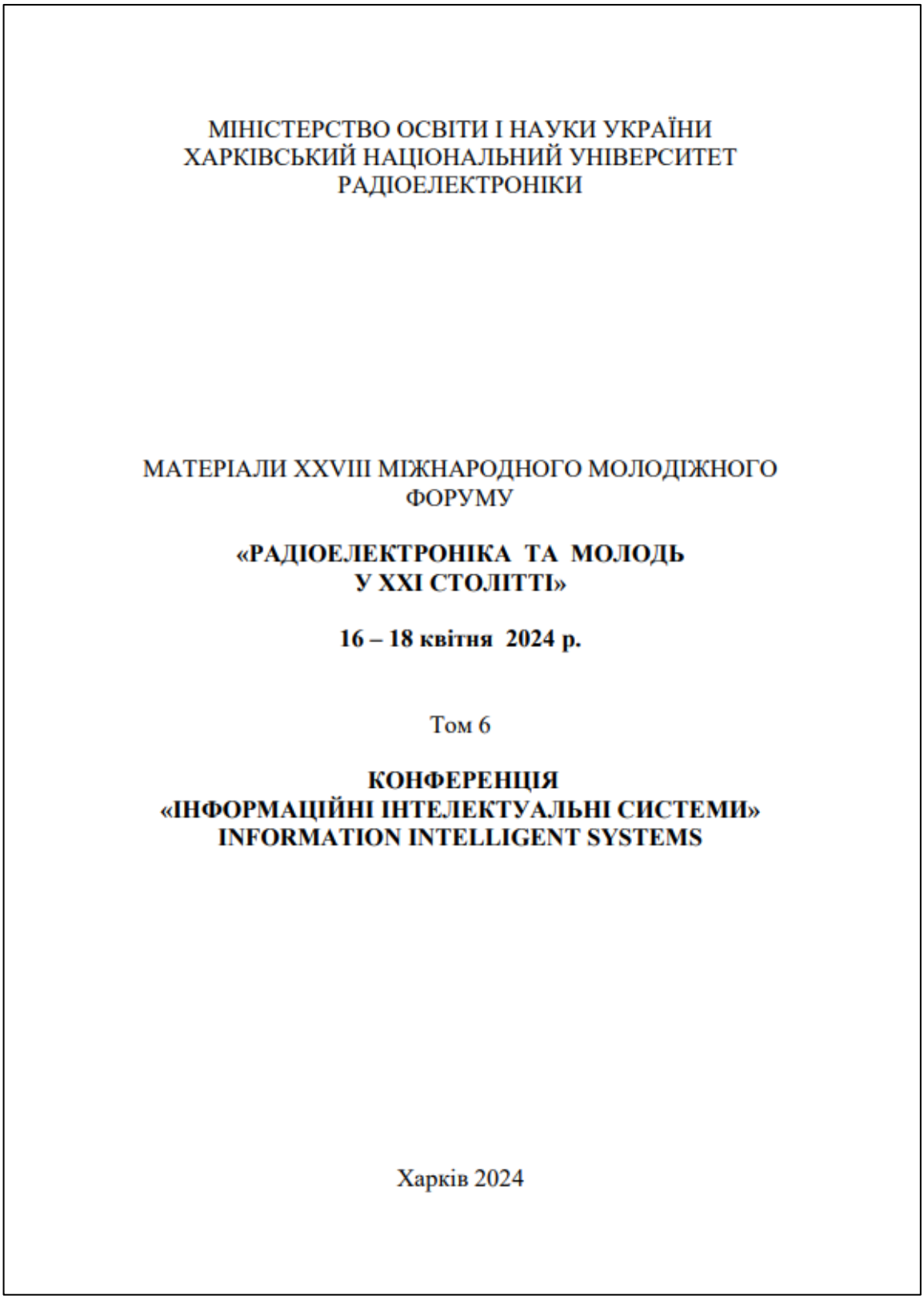
Голова оргкомітету Форуму Ігор РУБАН

18 квітня 2024 р.

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

ДОДАТОК И

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

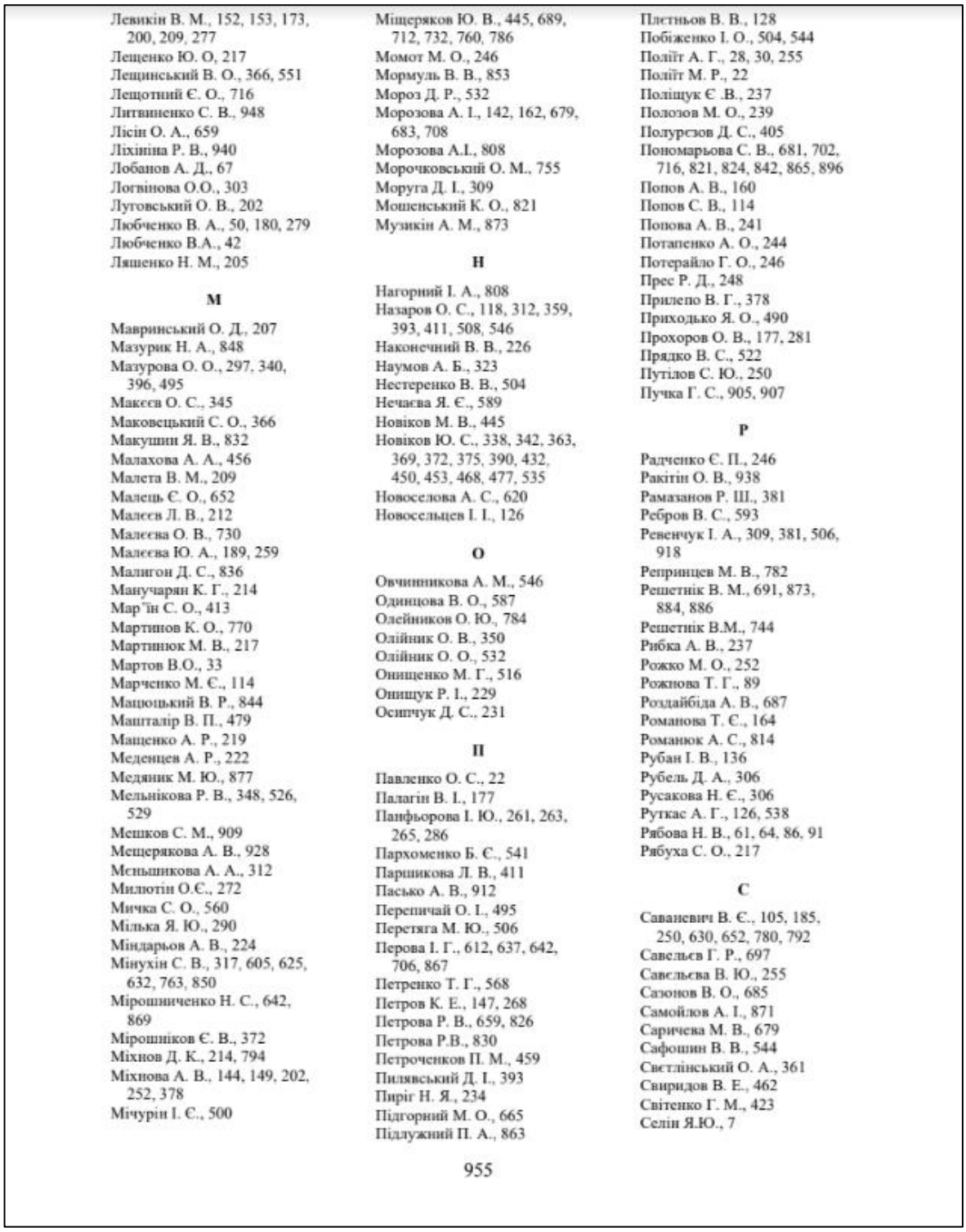


Рисунок И.2 – Алфавітний покажчик (знімок екрана виконано самостійно)

УДК 004.514:004.946

БАЛАНСУВАННЯ ЧАСУ ЩОДО СКЛАДНОСТІ ІГРОВОЇ СИТУАЦІЇ В MULTIPLAYER PARTY GAMES, КЛЮЧОВІ АСПЕКТИ ДЛЯ СТВОРЕННЯ ЗАХОПЛИВОГО ІГРОВОГО ПРОЦЕСУ

Мірошніков Є. В.
Науковий керівник – ст. викл. Новіков Ю. С.
Харківський національний університет радіоелектроніки, каф. ПІ
м. Харків, Україна
e-mail: yehor.miroshnikov@nure.ua

This research paper explores complexity and time management in multiplayer party games and examines their impact on player experience and overall game dynamics. It analyzes various aspects of game design, such as timers and unique gameplay structures, in order to identify general principles for achieving balance. Ensuring a balanced distribution of time between different aspects of gameplay is a key element of game design. The research aims to identify how to optimally manage time in games, providing a sense of challenge that maintains player interest and preserves the integrity of the gameplay.

Баланс гри у multiplayer party games зазвичай розуміється як запровадження певного рівня справедливості для гравців. Це включає в себе налаштування складності, умов виграшу-програшу, ігрових станів, балансування економіки тощо, щоб вони працювали в тандемі один з одним [1]. Правильна рівновага забезпечує чесну конкуренцію, підтримує інтерес гравців та цілісність гри. Аналіз тонкощів міні-ігр розкриває системний підхід до балансу, що ґрунтується на взаємодії таймерів, складності та унікальних ігрових структур.

Стандартним психологічним поясненням невдач у грі та викликів є теорія потоку Міхалі Чіксентміхалі, згідно з якою виклик певної діяльності формує вузький канал, в якому гравець перебуває у стані привабливого потоку. Хоча теорія потоку припускає, що гравець може коливатися між тривогою і нудьгою, вона створює банальну проблему, яка полягає в тому, що стандартна ілюстрація передбачає плавне збільшення складності з плином часу [2]. Ноа Фальштейн уточнив це твердження, сказавши, що складність гри повинна змінюватися хвилеподібно – іноді гра повинна бути трохи легшою, іноді трохи складнішою, і що нерівномірність призводить до задоволення, як показано на рисунку 1. Нерегулярне зростання складності збільшує ймовірність того, що гравець зазнає як невдач, так і успіхів [3].

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

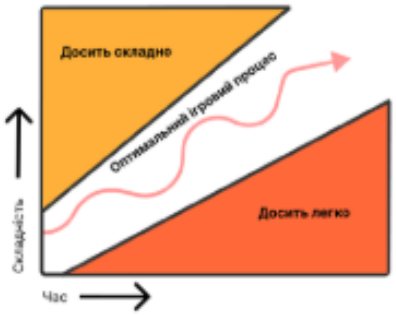


Рисунок 1 – Кращий канал потоку

«Turing Test»

У міні-грі «Turing Test» розглядається сценарій для одного студента та трьох професорів. Існують три раунди з окремими питаннями для взаємодії. Студент, вільно формулює питання для професорів, ініціюючи процес гри (див. рис. 2 – діаграма «Turing Test», легенда «1»). Далі професори повинні відповісти на запитання, що додає рівня складності запитань (див. рис. 2 – діаграма «Turing Test», легенда «2»). На завершальному етапі, студент ретельно аналізує питання і відповіді для ідентифікування штучного інтелекту серед професорів (див. рис. 2 – діаграма «Turing Test», легенда «3»).

Таким чином, забезпечення збалансовоного відведеного часу між складністю поставлених запитань, наданими відповідями та їх подальшим аналізом має першорядне значення.

«Brain Knights»

У міні-грі «Brain Knights» максимальна кількість гравців формує дві команди по чотири учасники кожна. Початковий раунд складається з двох підраундів і триває 90 секунд. Наступний триває 141 секунду, з яких 47 секунд виходить на одне питання.

Кожен раунд має різний рівень складності запитань. Перший розрахований на бліц-опитування з простими запитаннями, де капітан обирає гравця зі своєї команди, який найкраще орієнтується у темі (див. рис. 2 – діаграма «Brain Knights», легенди «1» та «2»). У другому, навпаки, застосовується колективний підхід, що дозволяє командам спільно обмірковувати відповіді на більш складні запитання (див. рис. 2 – діаграма «Brain Knights», легенди «1» та «3).

Гра забезпечує баланс за допомогою помірної складності раундів і достатнього часу на відповіді та обговорення. Капітан команди має достатньо часу для вибору відповідача та теми.

«Skibidy Party»

Міні-гра «Skibidy Party» вміщує до восьми гравців. Наприкінці кожного раунду призначений суддя, який розподіляє місця на основі

373

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

отриманих ситуацій та відповідних «мемів», тим самим отримуючи різний рівень складності (див. рис. 2 – діаграма «Skibidy Party», легенда «1»). Перед цим кожен гравець має завдання вибрати відповідний «мем» до ситуації (див. рис. 2 – діаграма «Skibidy Party», легенда «2»).

Ретельне впровадження таймерів у міні-гру слугує ключовим організаційним інструментом, що надає гравцям та судді необхідний час для розсудливого вибору «мемів».

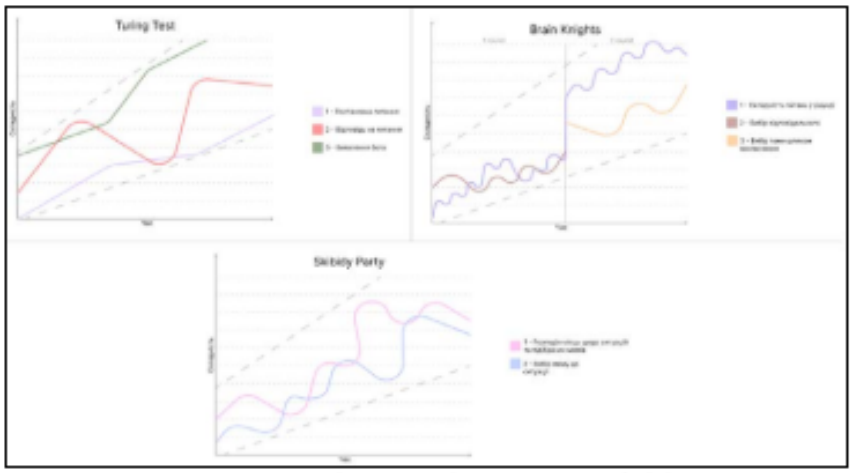


Рисунок 2 – Діаграма балансу для міні-ігр

Дослідження в міні-іграх вказує, що системний підхід до часових факторів і різноманітності завдань грає ключову роль у забезпеченні справедливої конкуренції, привертає увагу гравців та підтримує цілісність гри. В кожній міні-грі вдалося досягти збалансованого розподілу часу відповідно до кожної ігрової ситуації. Таким чином, вивчення геймплею в міні-іграх розкрило, що врахування часових аспектів та різноманітності завдань не лише підсилює конкуренцію, але й створює унікальні та захоплюючі моменти, що глибоко впливають на ігровий досвід гравців.

Список використаних джерел:

1. Contributors to Wikimedia projects. Game balance – Wikipedia. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Game_balance (дата звернення: 04.03.2024).
2. Juul J. Fear of Failing? The Many Meanings of Difficulty in Video Games. Jesper Juul. URL: https://www.jesperjuul.net/text/fearoffailing/ (дата звернення: 04.03.2024).
3. Game Balance: A Pivotal Issue in Game Design. Innovecs Games. URL: https://www.innovecsgames.com/blog/game-balance-a-critical-issue-in-designing-top-titles/ (дата звернення: 04.03.2024).

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

ДОДАТОК К

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

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