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. Back-end частина, інтеграція AI до системи
(тема)

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

Калініченко О. Ю.
(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 Калініченко Олександру Юрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Ігровий програмний застосунок у жанрі Multiplayer Party Games.Back-end частина, інтеграція AI до системи
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)_____ Калініченко О. Ю.
(підпис)

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

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, 171 стор., 28 рис., 9 додатків, 1 табл., 12 джерел.

БАГАТОКОРИСТУВАЦЬКІ ІГРИ ДЛЯ ВЕЧІРОК, ІГРОВИЙ ПРОГРАМНИЙ ЗАСТОСУНОК, ІНТЕГРАЦІЯ ШТУЧНОГО ІНТЕЛЕКТУ, AXIOS, BACK-END, GPT TEXT-DAVINCI-003, NESTJS, NODE.JS, TYPESCRIPT, WEBSOCKET

Об’єкт розробки – back-end частина ігрового програмного застосунку у жанрі multiplayer party games.

Мета розробки – розробити back-end частину системи для ігрового програмного застосунку у жанрі multiplayer party games та інтегрувати до неї штучний інтелект.

Метод рішення – середовище розробки Visual Studio Code, програмна платформа Node.js, фреймворк NestJS, бібліотеки Axios та WebSocket, мова програмування TypeScript.

У результаті розробки було створено back-end частину системи та інтегровано штучний інтелект у ігровий програмний застосунок у жанрі multiplayer party games, який містить три міні-гри. У межах кваліфікаційної роботи було розроблено back-end частину системи, яка охоплює логіку двох багатокористувацьких ігор для вечірок наступних жанрів: змагальна вікторина та соціальна дедуктивна гра.

MULTIPLAYER PARTY GAMES, GAME SOFTWARE, INTEGRATION OF ARTIFICIAL INTELLIGENCE, AXIOS, BACK-END, GPT TEXT-DAVINCI-003, NESTJS, NODE.JS, TYPESCRIPT, WEBSOCKET

The object of development is the back-end part of a gaming software application in the genre of multiplayer party games.

The purpose of the development is to develop the back-end part of the system for a gaming software application in the genre of multiplayer party games and integrate artificial intelligence into it.

The solution method used was the Visual Studio Code development environment, the Node.js software platform, the NestJS framework, the Axios and WebSocket libraries, and the TypeScript programming language.

As a result of the development, the back-end part of the system was created and artificial intelligence was integrated into a gaming software application in the genre of multiplayer party games, which contains three mini-games. As part of the qualification work, the back-end part of the system was developed, which includes the logic of two multiplayer party games of the following genres: a competitive quiz and a social deduction game.

Я, Калініченко Олександр Юрійович, 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.Back-end частина, інтеграція AI до системи», 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

MVP – Most Valuable Player

API – Application Programming Interface

OpenAI – Open Artificial Intelligence

JSON – JavaScript Object Notation

GPT – Generative Pre-trained Transformer

ШІ – Штучний Інтелект

ПЗ – Програмне Забезпечення

UML – Unified Modeling Language

HTTP – HyperText Transfer Protocol

UUID – Universally Unique Identifier

ВСТУП

Зростаюча популярність колективних ігор свідчить про те, що сучасні користувачі шукають способи спільного проведення часу та розваг. У цьому контексті розробка ігрового програмного застосунку у жанрі multiplayer party games набуває особливої актуальності та стає пріоритетним завданням. Ігри цього жанру сприяють активній взаємодії та розвагам у колі друзів та знайомих, створюючи веселу та захоплюючу атмосферу. Інтеграція штучного інтелекту, створення захопливого геймплею та налаштування логіки обробки міні-ігор на back-end частині системи стає актуальною задачею, результат якої не лише відповідатиме сучасним вимогам і очікуванням гравців, але й дозволить конкурувати на ринку ігрових продуктів у цьому жанрі.

Темою кваліфікаційної роботи є розробка back-end частини та інтеграція штучного інтелекту в ігровий програмний застосунок у жанрі multiplayer party games. Основне завдання полягає у створенні ефективної системи, що сприятиме взаємодії користувачів з ігровим програмним застосунком, забезпечуючи підвищений рівень зацікавленості у грі.

Метою роботи є розробка back-end частини ігрового програмного застосунку, що буде відповідати за обробку логіки міні-гри у жанрі змагальна вікторина, а також інтеграція штучного інтелекту до соціально-дедуктивної міні-гри для створення реалістичного ігрового супротивника. Для досягнення цих цілей використовуватимуться такі технології, як середовище розробки Visual Studio Code, програмна платформа Node.js, фреймворк NestJS, бібліотеки Axios та WebSocket, а також мова програмування TypeScript.

Цей ігровий застосунок призначений для компанії знайомих і друзів, які бажають провести вільний час в онлайн грі. Будь-який комп’ютер або консоль з доступом до Інтернету та веб-браузера може бути хостом, а інші учасники можуть використовувати веб-браузери на своїх смартфонах чи інших пристроях як контролери. Це робить процес гри цікавим, зручним і доступним для всіх.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Multiplayer party games – це жанр ігор, призначений для гри групами людей у неформальній обстановці, зазвичай на вечірках, зборах друзів та знайомих. Ці ігри орієнтовані на командну взаємодію, змагання або просто розвагу, мають прості правила та механіки, що дозволяє легко залучати гравців усіх рівнів. Останнім часом спостерігається збільшення популярності ігор цього жанру серед різних користувачів. Ця зацікавленість пояснюється зростанням соціальної активності в цифровому середовищі та здатністю об’єднувати людей для спільного проведення часу, а також для створення дружньої атмосфери як у командних, так і у змагальних іграх. Перспективи розвитку сфери ігор у жанрі multiplayer party games величезні, і продукти, які пропонують високу якість геймплею, оригінальність та можливості для спільного часопроведення, матимуть великий попит на ринку. Розумний підхід до розробки таких ігор, з урахуванням поточних тенденцій та потреб споживачів, може відкрити шлях до успіху та довгострокової популярності серед широкого кола гравців. Майбутній потенціал розвитку також свідчить про подальше зростання цього сегменту ринку ігрової індустрії. З підвищенням доступності технологій, розвитком штучного інтелекту та розширенням можливостей віртуальної реальності очікується ще більше інноваційних рішень, які матимуть важливе значення як для звичайних гравців, так і для розробників.

На ринку спостерігається різноманітність ігрових програмних застосунків у жанрі multiplayer party games, які варіюються за стилями, геймплеєм, технічними можливостями та механіками. Конкуренція дуже висока, оскільки існують багато ігрових розробників, які створюють ігри для різних платформ та аудиторій. Для досягнення конкурентоспроможності важливо постійно підтримувати свій продукт, розширювати його, періодично вносити щось нове, дотримуватись поточних тенденцій розвитку, а також забезпечити різні режими гри. Таким чином, ключовим фактором успіху є розробка ігрового програмного застосунку, який не

лише відповідає сучасним вимогам гравців, а й здатний привертати увагу нових користувачів своєю оригінальністю та якістю геймплею.

Ігровий програмний застосунок у жанрі multiplayer party games складається з клієнтської та серверної частин. Серверна частина відповідає за забезпечення стабільної роботи мережі, обробку, управління та зберігання даних ігрової сесії. Клієнтська частина надає зручний функціонал у керуванні, відображенні, передачі даних на сервер та обробці отриманих даних. Клієнтська частина поділяється на хост та контролер.

Хост – комп'ютер або консоль, на якому відображається головний екран гри та на якому запущений основний ігровий процес через веб-браузер. Він відповідає за обробку даних сервера, відображення та керування основним ігровим процесом, а також надає можливість обрати одну з розроблених міні-ігор та створює приватну ігрову кімнату, до якої можуть приєднатися інші контролери, щоб розпочати ігрову сесію.

Контролер – смартфон або інший пристрій, який використовується для відображення поточного етапу гри, а також для керування та взаємодії з грою через веб-браузер. Перед початком гри, після створення кімнати на хості, гравці вводять її унікальний код або переходять за посиланням, щоб приєднатися до ігрової сесії.

Цільова аудиторія ігор у жанрі multiplayer party games охоплює людей різного віку, які насолоджуються соціальною взаємодією та активним відпочинком у компанії друзів та знайомих. Здебільшого це чоловіки, віком від 14 до 35 років, які цінують конкуренцію, стратегічну логіку та тактику. Наявність легких механік, які не потребують великого досвіду, дозволяє привернути навіть тих, хто раніше не мав досвіду в іграх. Таким чином, продукт спрямований на широкий спектр гравців, які шукають якісний, нескладний та соціально захоплюючий геймплей.

Мета ігрового програмного застосунку у жанрі multiplayer party games полягає у створенні якісного та захоплюючого ігрового досвіду для користувачів, які хочуть спільно провести час та розважитися. Основними цілями цієї системи є:

– розробка цікавого та різноманітного геймплею, який сприятиме активній взаємодії між гравцями та надасть їм задоволення від гри;

– розробка програмного застосунку, який буде легко доступний для користувачів у будь-який час та з будь-якого пристрою з підключенням до Інтернету та наявністю веб-браузера;

– забезпечення надійної мережевої взаємодії між гравцями для плавної та безперебійної гри в режимі реального часу з можливістю перепідключення до розпочатої ігрової сесії;

– реалізація простих та цікавих механік, що стимулюватимуть не тільки конкуренцію в міні-іграх, але й навички командної співпраці, забезпечуючи доступність гри для гравців різного рівня;

– забезпечення доступності та зручності для широкого кола користувачів за допомогою інтуїтивно зрозумілого інтерфейсу та впровадження локалізації до застосунку.

Таким чином, визначена мета ігрового програмного застосунку полягає у створенні продукту, який привертає увагу широкого кола гравців, забезпечує активну участь їх у грі, має налагоджену мережеву роботу та є максимально доступним для всіх користувачів системи.

Фрагменти предметної області з розробки ігрового програмного застосунку у жанрі multiplayer party games включають:

а) для хоста:

1) створення та налаштування ігрових сесій, включаючи вибір міні-гри та параметрів гри;

2) керування ігровою сесією та відображення інформації про поточний етап гри;

3) обробка отриманих даних з сервера;

б) для контролера:

1) підключення до існуючих ігрових сесій за допомогою унікального ключа кімнати;

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

3) обробка отриманих даних з сервера;

в) для сервера:

1) зберігання та управління ігровими даними;

2) обробка отриманих даних від клієнту;

3) забезпечення стабільної роботи мережі та підтримка з’єднання між хостом і контролерами.

Ці фрагменти предметної області включають перелік завдань, що виконують управління даними, їх зберігання, обробку та виведення, а також забезпечення взаємодії користувачів системи між собою та з сервером.

Інформаційні потреби предметної області з розробки ігрового програмного застосунку у жанрі multiplayer party games включають:

– отримання стислої інформації про міні-гру перед її вибором на хості;

– отримання інформації про гру при підключенні до неї за допомогою унікального ключа кімнати на контролері;

– отримання інформації про поточний стан гри на хості та контролері;

– отримання інформації про дії, які потрібно виконати в певний момент гри на хості та контролері;

– отримання інформації про кінцевий результат гри на хості та контролері.

Інформаційні потреби користувачів у жанрі multiplayer party games забезпечують доступ до необхідної інформації про міні-ігри, хід геймплею, взаємодію з іншими гравцями, результати та досягнення, що сприяє формуванню приємного та зрозумілого ігрового досвіду.

Розглянемо декілька існуючих аналогів, подібних до розроблюваного ігрового програмного застосунку у жанрі multiplayer party games.

Перший аналог – серія ігор The Jackbox Party Pack, розроблена студією Jackbox Games. Одна з основних відмінностей The Jackbox Party Pack від інших ігор у цьому жанрі полягає в асиметричному геймплеї та унікальному форматі.

Після завантаження гри на свій пристрій гравець може обрати одну з запропонованих міні-ігор через головне меню (див. рис. 1.1). Крім міні-ігор головне меню містить елементи навігації, налаштування, документацію та пояснення до кожної міні-гри.

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

Після вибору міні-гри, гра створює приватну кімнату з унікальним чотирьохзначним кодом (див. рис. 1.2). На цьому екрані відображається меню очікування підключення інших гравців через веб-браузер.

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

Через свої пристрої гравці відкривають веб-браузер та переходять на веб-сайт https://jackbox.tv/, де їх зустрічає меню для підключення до приватної кімнати (див. рис. 1.3). Тут можна переглянути загальну інформацію про компанію та їх продукти, побачити список доступних пакетів, ввести код кімнати та свій нікнейм, а потім натиснути кнопку «Play», щоб підключитися до ігрової сесії.

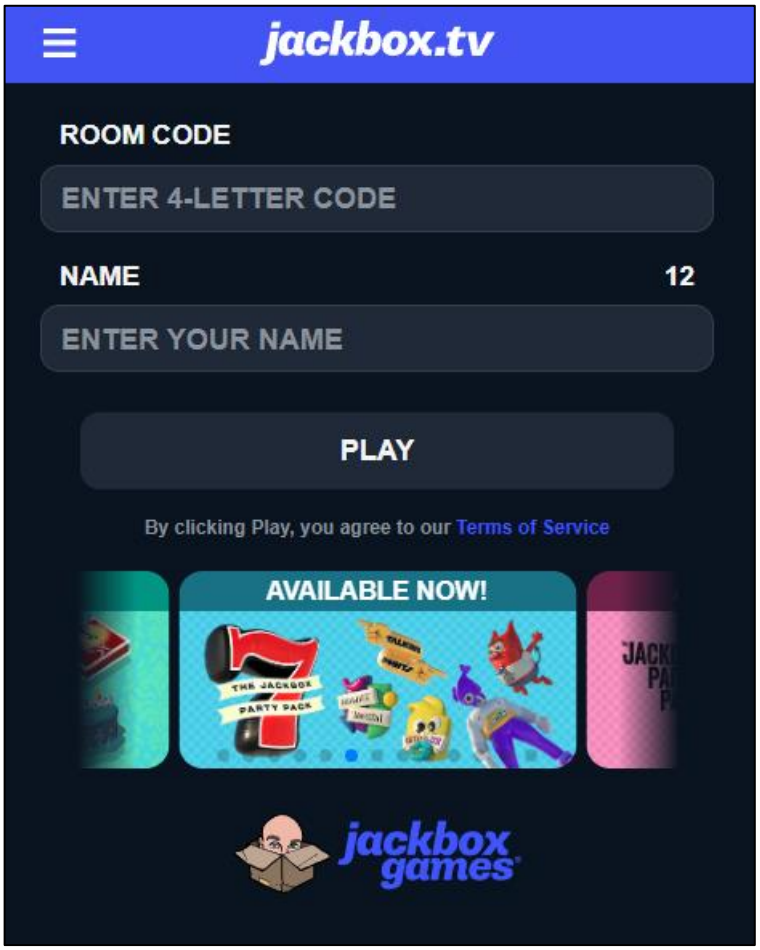


Рисунок 1.3 – Меню підключення до приватної кімнати на веб-сайті https://jackbox.tv/ (знімок екрана виконано самостійно)

На прикладі міні-гри Trivia Murder Party, на екрані хоста (див. рис. 1.4) відображаються гравці, що беруть участь у раунді, а також питання на яке їм потрібно дати відповідь. Питання додатково озвучує диктор.

Рисунок 1.4 – Екран хоста The Jackbox Party Pack з запущеною міні-грою Trivia Murder Party (знімок екрана виконано самостійно)

Після оголошення етапу відповіді на питання на екрані контролерів гравців (див. рис. 1.5) повторюється питання, і виводяться варіанти відповідей на нього.

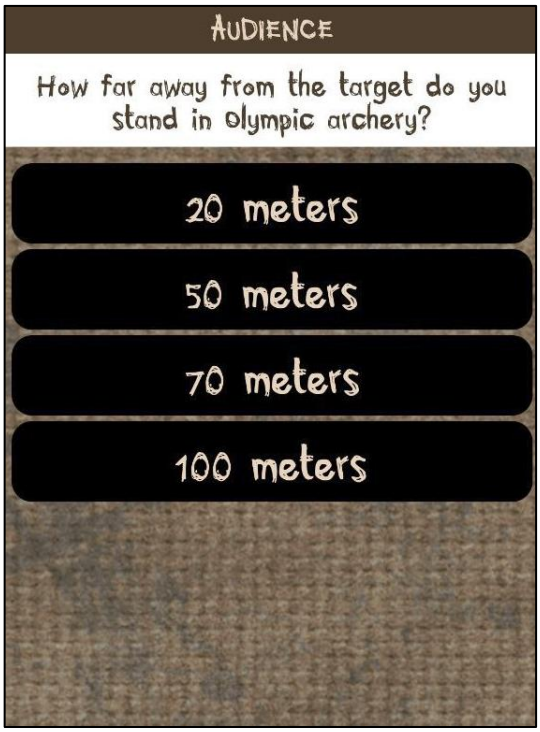


Рисунок 1.5 – Екран етапу відповіді користувачів на веб-сайті https://jackbox.tv/ (знімок екрана виконано самостійно)

До переваг цієї гри можна віднести наступне:

– інформативність;

– простота використання;

– різноманітність міні-ігор;

– приємні анімації та інтерфейс користувача;

– професійно озвучені коментарі та правила міні-ігор;

– регулярні оновлення контенту міні-ігор у вже створених пакетах.

До недоліків цієї гри можна віднести наступне:

– деякі ігри можуть мати обмеження у перекладі;

– незважаючи на те, що окремі ігри коштують дешевше, пакети можуть бути досить дорогими;

– на відміну від ігор, що доступні через веб-браузер, гра вимагає завантаження та встановлення на персональний комп’ютер або ігрову консоль.

Другий аналог – це онлайн-гра Gartic Phone, розроблена студією Onrizon Social Games. Основна відмінність Gartic Phone від інших ігор у жанрі multiplayer party games, полягає в її креативному та унікальному підході до ігрового процесу, де кожен учасник робить свій внесок у створення історії або малюнка, що створює веселі, унікальні та несподівані ігрові історії.

Після відкриття веб-додатку https://garticphone.com/, користувачі потрапляють на екран головного меню (див. рис. 1.6). Тут можна переглянути інформацію про соціальні мережі розробників, правила гри у вигляді слайдеру з анімаціями, налаштувати мову, ввести свій нікнейм, обрати аватар та розпочати ігрову сесію натиснувши кнопку «Start». Увійти можна як анонімний користувач, або авторизуватися у системі через месенджер Discord, або через відеострімінговий сервіс Twitch.

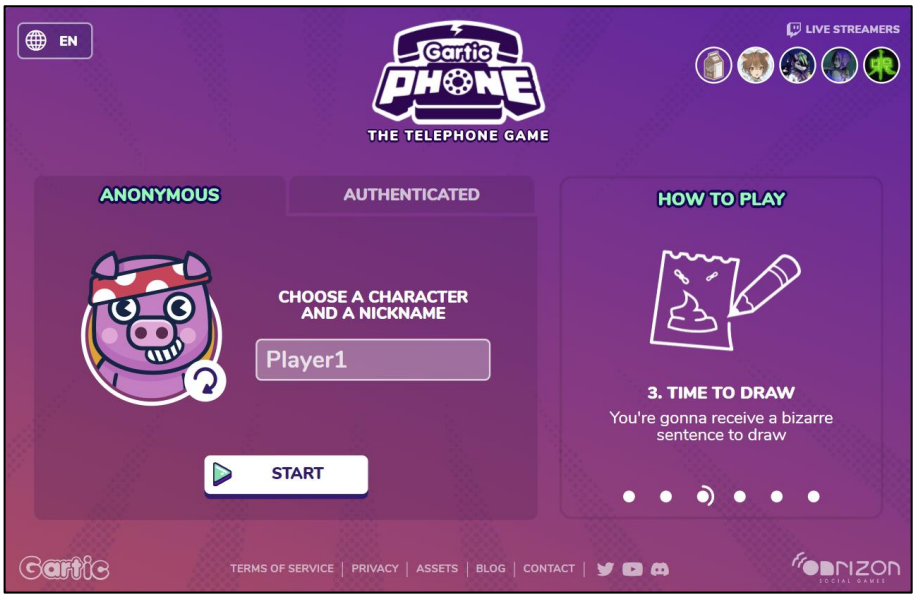


Рисунок 1.6 – Меню підключення до приватної ігрової кімнати на веб-сайті https://garticphone.com/ (знімок екрана виконано самостійно)

Після створення приватної кімнати, користувач потрапляє на екран (див. рис. 1.7), де він може налаштувати кількість гравців, обрати один з запропонованих режимів гри, скопіювати посилання за допомогою кнопки «Invite», через яке інші гравці зможуть доєднатися до кімнати, а також розпочати гру за допомого кнопки «Start».

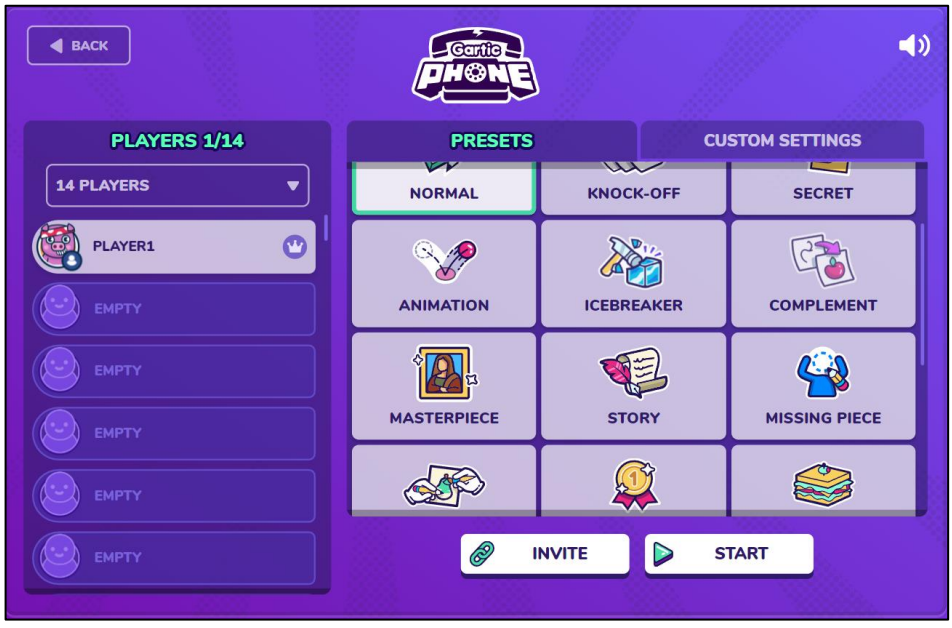


Рисунок 1.7 – Створена приватна кімната на веб сайті https://garticphone.com/ (знімок екрана виконано самостійно)

В залежності від обраного режиму, який загалом можна розділити на малювання або написання історій, користувачі бачать перед собою екран (див. рис. 1.8), на якому описане поточне завдання, а також або наведені приклади його виконання або надаються усі необхідні інструменти для його виконання.

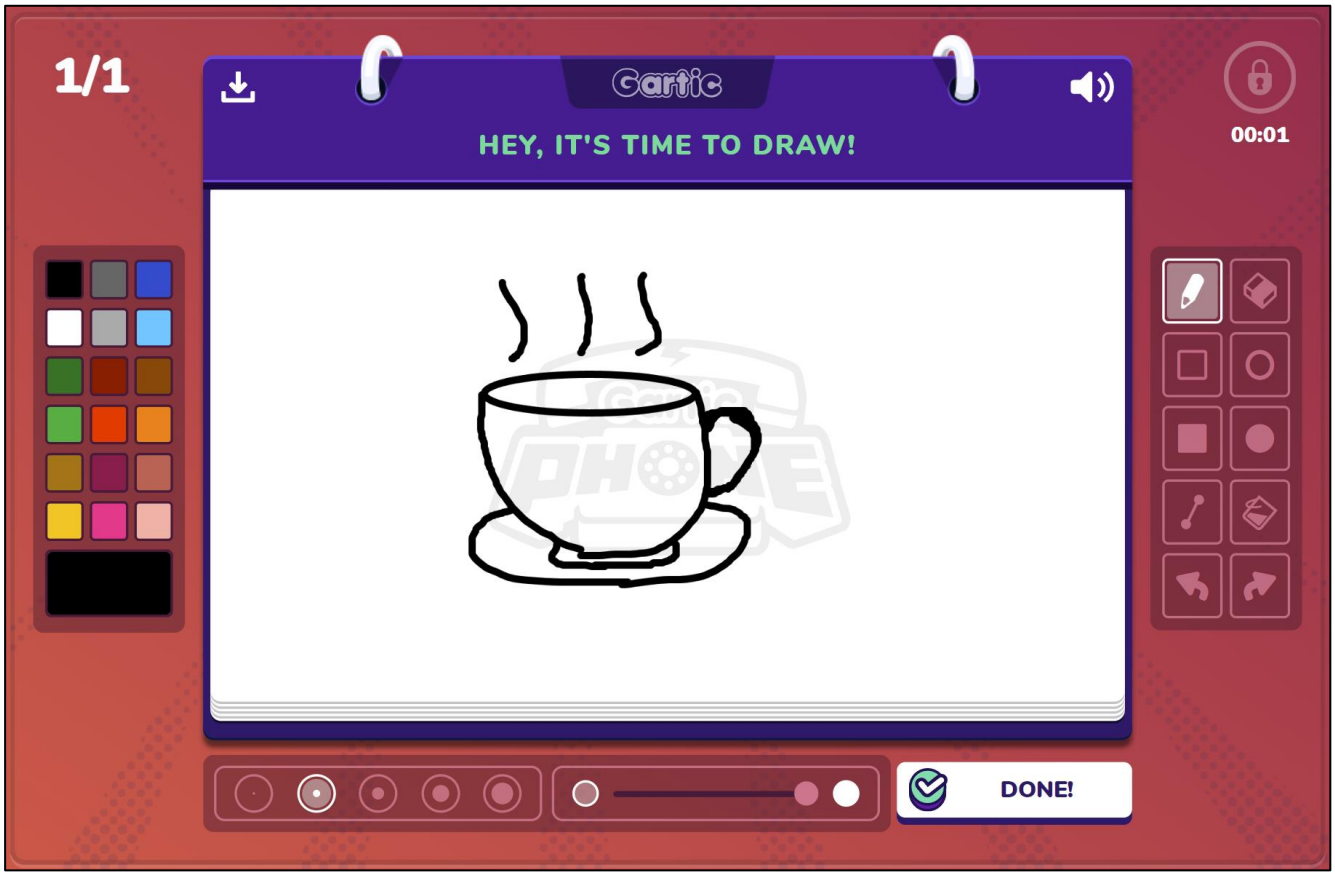


Рисунок 1.8 – Екран з завданням на веб сайті https://garticphone.com/ (знімок екрана виконано самостійно)

Після виконання всіма користувачами отриманих завдань настає етап підсумку (див. рис. 1.9), де виводяться результати гри: створений альбом кожного гравця, де кожен учасник так чи інакше вплинув на отриманий фінальний результат або власна створена історія, що доповнювалась кожним учасником гри.

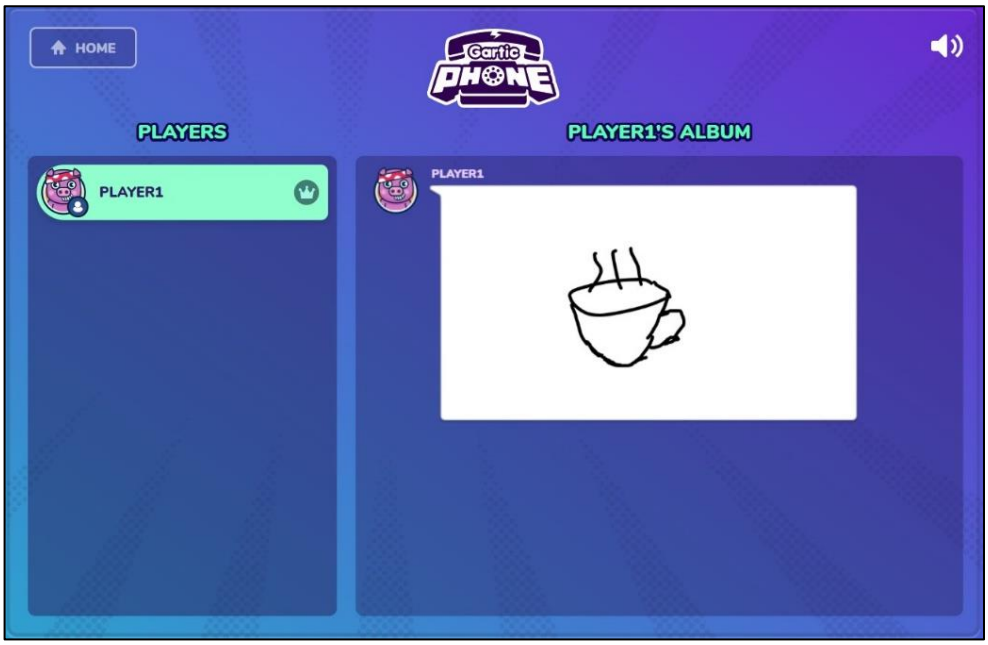


Рисунок 1.9 – Екран підсумку гри на веб сайті https://garticphone.com/ (знімок екрана виконано самостійно)

До переваг цієї гри можна віднести наступне:

– інформативність;

– простота використання;

– приємний інтерфейс користувача;

– безкоштовність;

– немає необхідності у завантажені гри до свого пристрою;

– можливість проведення гри у досить великій компанії гравців;

– можливість збереження створених малюнків та історій в кінці гри;

– велика кількість підтримуваних перекладів.

До недоліків цієї гри можна віднести наступне:

– однотипність режимів, що включають у себе написання історій або малювання;

– інструменти редагування обмежені, що може ускладнити творчість для деяких користувачів;

– інтегрована озвучка, в окремі моменти, може звучати недосконало.

Після проведеного аналізу існуючих аналогів можна зробити висновок, що для забезпечення конкурентоспроможності необхідно врахувати максимум переваг

та мінімум недоліків, а також за можливістю створити щось унікальне, таке, що інші продукти ще не пропонували для своєї аудиторії.

1.2 Виявлення та вирішення проблем

З розвитком інформаційних технологій та поширенням доступу до Інтернету, ігрова індустрія постійно знаходить нові шляхи для залучення гравців та створення захопливого ігрового досвіду. У такому контексті жанр multiplayer party games набуває все більшої популярності, спрямовуючись на соціальну взаємодію та активне спільне проведення часу компанії друзів та знайомих у віртуальному просторі.

Зі зростаючою популярністю ігор у жанрі multiplayer party games зростає потреба у вдосконаленні ігрових програмних застосунків для забезпечення задоволення користувачів та підвищення конкурентоспроможності на ринку. Виявлення та ефективне вирішення проблем стає ключовим завданням для забезпечення якісного та захоплюючого ігрового досвіду для гравців. Наявність програмного застосунку, який може запускатися у веб-браузері без необхідності його завантаження на власний пристрій, дозволить користувачам швидко та без зайвих зусиль приєднатися до гри у будь-який момент часу. Прості механіки забезпечать комфортну гру як для новачків, так і для досвідчених гравців. Різноманітний геймплей та жанри міні-ігор дозволять привернуть увагу широкого кола користувачів. Інтеграція штучного інтелекту дозволить не лише створити унікальні ігрові сценарії та привернути увагу ще більшої аудиторії, але й дозволить продукту виокремитися на фоні конкурентів завдяки інноваційному підходу до геймінгу. Постійні оновлення та підтримка продукту збережуть зацікавленість гравців та забезпечать його актуальність на ринку.

Ефективне вирішення цих проблем дозволить покращити якість ігрового програмного застосунку, збільшити конкурентоспроможність на ринку ігор у жанрі

multiplayer party games, а також сприятиме зміцненню соціальних зв’язків та комунікації між гравцями, роблячи віртуальні ігри ще цікавішими.

1.3 Постановка задачі

Для вирішення та задоволення потреб користувачів, які прагнуть соціальної взаємодії та проведення активного спільного часу у компаній друзів та знайомих у віртуальному просторі, потрібно розробити ігровий програмний застосунок у жанрі multiplayer party games, який на першому етапі розробки міститиме три міні-гри, які можна обрати в головному меню хоста: Brain Knights, Turing Test та Skibidy Party. Однією з ключових складових процесу буде розробка back-end частини системи та інтеграція штучного інтелекту.

Brain Knights – це командна змагальна вікторина, розрахована на участь від чотирьох до восьми гравців і складається з двох етапів. Перший етап передбачає проведення серії раундів. Кількість раундів визначається максимальною кількістю гравців у одній з двох команд, помноженою на два, щоб кожен учасник прийняв участь у першому етапі принаймні двічі. У кожному раунді випадково вибирається тема, після чого капітан кожної команди обирає одного гравця, який буде відповідати на запитання з обраної теми. Кожне питання має чотири варіанти відповідей, одна із яких є правильною, а інші – ні. Система веде облік правильних відповідей, і в кінці раунду учасник, який набрав найбільшу кількість правильних відповідей, приносить своїй команді одне зароблене очко. Якщо кількість правильних відповідей однакова, то обидва учасники приносять своїм командам по одному зробленому очку. Під кінець першого етапу визначається MVP гри – гравець, який надав найбільшу кількість правильних відповідей. Другий етап – це командна вікторина, в якій беруть участь всі члени команди. Етап складається з трьох раундів. Перед кожним раундом капітани, чергово обираючи, виключають зі списку чотири теми. Питання стають складнішими, а командам надається значний

час на обговорення запитання, що має одну правильну відповідь. За кожну правильну відповідь команда отримує одне очко до загального рахунку. Після другого етапу визначається команда-переможець, яка набрала найбільшу кількість балів. У випадку однакової кількості балів визначається нічия.

Turing Test – це командна соціальна дедуктивна гра, розрахована на участь від чотирьох до восьми гравців та складається з трьох раундів. Гравці розподіляються на команди студентів і професорів за таким співвідношенням: чотири гравці – один студент та три професори; п’ять гравців – два студенти та три професори; шість гравців – два студенти та чотири професори; сім гравців – три студенти та чотири професори; вісім гравців – три студенти та п’ять професорів. Мета студентів – визначити бота (штучний інтелект) серед професорів, тоді як команда професорів має заплутати команду студентів, маскуючись під штучний інтелект. Гравці з команди студентів починають гру, кожен із них ставить запитання, на яке професори та бот повинні відповісти. Потім професори дають свої відповіді. Після цього бот отримує запитання та варіанти відповідей, і намагається замаскуватися під професорів. Потім відбувається голосування, під час якого студенти повинні визначити, яка з відповідей належить боту. Якщо вони правильно впізнали бота, то отримують два очки до загального рахунку, а якщо ні – одне очко отримує команда професорів. Перемагає та команда, яка набрала більше очок. Якщо ж очки розподілилися порівну, гра закінчується в нічию.

Skibidy Party – це гумористична карткова гра, розрахована на участь від чотирьох до восьми гравців, що складається з кількості раундів, рівної кількості учасників гри. Кожен раунд випадковим чином обирається суддя, який оцінює вибрані гравцями меми відповідно до випадково обраної ситуації. У кожному раунді звичайні гравці отримують по шість карток з мемами. Після того, як випадковим чином обирається ситуація із відповідної колоди карток, гравці вибирають найкумедніший мем, який вони мають на руках, і що на їхню думку найкраще підходить до ситуації. Потім суддя отримує список мемів, обраних гравцями, і розставляє їх по місцях від першого до третього. За перше місце

учасник отримує 1500 очок, за друге – 1000 очок, за третє – 500 очок. Перемагає той, хто набрав найбільшу кількість балів протягом усіх проведених раундів.

Ігровий програмний застосунок складається з двох компонентів: серверної та клієнтської частин, клієнтська частина включає в себе хост та контролер. Гра запускається або на комп’ютері, або на консолі, яка має доступ до Інтернету та веб-браузера, у якості хоста. На хості обирається одна з запропонованих міні-ігор, після чого створюється приватна ігрова кімната, що має свій власний унікальний код. Гравці за допомогою своїх смартфонів або інших пристроїв, які виступають контролерами, відкривають веб-додаток і вводять там свій нікнейм та чотирьохзначний код приватної кімнати. Перший гравець, який приєднується до кімнати, стає її лідером і може розпочати гру, коли набереться мінімальна кількість користувачів, які підтвердять свою готовність, оберуть аватари або введуть назву команди, якщо гра передбачає це.

У межах кваліфікаційної роботи на back-end частині системи необхідно розробити логіку міні-гри у жанрі змагальної вікторини Brain Knights та інтегрувати штучний інтелект до соціально-дедуктивної міні-гри Turing Test для створення реалістичного ігрового супротивника. Back-end частина повинна створювати приватні ігрові сесії, проводити ідентифікацію підключених гравців, обробляти та валідувати отримані дані з клієнта, зберігати дані сесії у своїй пам’яті, а після її завершення очищати їх.

Back-end частина системи повинна бути розроблена з використанням мови програмування TypeScript, програмної платформи Node.js та фреймворку NestJS [1-3]. Для забезпечення обміну даними між клієнтською та серверною частинами в реальному часі повинна бути використана бібліотека WebSocket, а для формування та відправлення запитів до API OpenAI повинна використовуватись бібліотека Axios [4-5]. Для налаштування гри буде використаний файл у форматі .json, що міститиме ідентифікатор гри, її назву, мінімальну та максимальну кількість гравців, а також заготовлений список тем, питань та відповідей українською та англійською мовами для забезпечення локалізації ігрового програмного застосунку.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Основною метою ігрового програмного застосунку у жанрі multiplayer party games, у межах кваліфікаційної роботи, є розробка та налаштування логіки міні-гри Brain Knights на back-end частині системи, а також інтеграція штучного інтелекту за допомогою API OpenAI до міні-гри Turing Test для створення реалістичного ігрового супротивника.

Back-end частина ігрового програмного застосунку у жанрі multiplayer party games повинна вирішувати наступні завдання:

– ідентифікація підключених користувачів: хост і контролер;

– управління приватними ігровими кімнатами;

– управління статусом підключення гравців до ігрової сесії;

– збереження даних ігрової сесії та їх видалення після її завершення;

– обробка дій користувачів, що взаємодіють з клієнтською частиною, відповідно до логіки розпочатої міні-гри.

Back-end частина ігрового програмного застосунку у жанрі multiplayer party games має наступні функціональні вимоги:

а) розробка та налаштування логіки міні-гри Brain Knights:

1) гравець повинен мати можливість підключитися, відключитися та перепідключитися до ігрової сесії;

2) система повинна надати першому підключеному гравцю до кімнати статус власника кімнати, а якщо він відключився, вона повинна передати цей статус наступному користувачу у списку;

3) власник кімнати повинен мати змогу розпочати ігрову сесію за умови підключення необхідного мінімуму гравців для гри та підтвердження їхньої готовності;

4) хост-користувач повинен мати можливість змінювати статус гри, зокрема ставити її на паузу та відновлювати ігровий процес;

5) у разі відключення хост-користувача від кімнати, система повинна відключити всіх інших гравців та очистити дані ігрової сесії;

6) система повинна виводити загальну інформацію про ігрову сесію для користувачів, а саме: код кімнати, список гравців, їхні дані, та час, що залишився до зміни екрану;

7) система повинна валідувати та обробляти отримані дані від клієнтської частини;

8) перед початком ігрової сесії система повинна створити чергу екранів для відображення з налаштуванням ролей, часу, викликаними функціями як перед зміною екрану, так і в момент його відображення;

9) після завершення гри, система повинна автоматично очистити дані ігрової сесії;

10) після того, як гравець увійшов у кімнату, де ігрова сесія ще не розпочалася, система повинна автоматично надати йому випадковий аватар;

11) система повинна надавати можливість гравцям змінити аватар;

12) система повинна надавати можливість гравцям змінити команду;

13) система повинна розподіляти гравців по командах, де менше користувачів, якщо їх рівна кількість або гравець перший заходить до кімнати, то його записує до червоної команди;

14) система повинна надавати роль капітана першим гравцям у командах;

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

16) система повинна надавати можливість капітану передати свою роль іншому гравцю;

17) система повинна надавати можливість капітану змінити назву команди;

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

19) система повинна визначати кількість раундів;

20) система повинна випадковим чином обирати тему для кожного раунду;

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

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

23) система повинна надавати можливість отримувати інформацію про вибрані гравцями відповіді та їх правильність;

24) система повинна виконувати підрахунок балів для команд у кінці кожного раунду;

25) система повинна надавати можливість отримувати інформацію про загальний рахунок балів для кожної команди;

26) система повинна рахувати кількість правильних відповідей кожного гравця;

27) система повинна визначати MVP гравців гри – тих, хто надав найбільшу кількість правильних відповідей;

28) система повинна надавати можливість капітанам під час другого етапу гри почергово обирати теми, які необхідно вилучити зі списку;

29) система повинна визначати команду, яка перемогла у грі;

30) система повинна розподіляти досягнення гравців, які вони здобули протягом ігрової сесії;

б) інтеграція штучного інтелекту до міні-гри Turing Test:

1) налаштування бота на надання природних відповідей;

2) налаштування передачі запитань студентів та відповідей професорів до промпту для маскування бота;

3) налаштування бота так, щоб він мав змогу адаптувати свою відповідь до будь-якої ігрової ситуації;

4) повернення випадкової загальновживаної відповіді з попередньо підготовленого списку у випадку виникнення помилки під час запиту до API OpenAI.

Back-end частина ігрового програмного застосунку у жанрі multiplayer party games має наступні нефункціональні вимоги:

– контент ігор, що знаходиться у статичних файлах формату .json, повинен підтримувати українську та англійську мови;

– кожен отриманий запит від клієнтської частини повинен логуватися для відстеження помилок та збоїв системи;

– налаштування промпту для отримання відповіді від ШІ повинно бути перевірене, щоб гарантувати коректність і відповідність очікуваним результатам.

Інтеграція штучного інтелекту, а також реалізація та налаштування логіки міні-гри Brain Knights, буде виконана на back-end частині програмного застосунку з використанням мови програмування TypeScript, програмної платформи Node.js та фреймворку NestJS. Інтеграція ШІ буде здійснена за допомогою API компанії OpenAI, яке надає доступ до використання моделі GPT Text-Davinci-003, що найбільш підходить для відтворення природного спілкування та реалістичної взаємодії з користувачами. Технології, які будуть використовуватися: Node.js, NestJS, Axios, Class Validator, WebSocket та TypeScript.

Розробка ігрового програмного застосунку буде здійснюватися у кілька етапів:

а) підготовчий етап:

1) формулювання та узгодження системних вимог;
2) розробка програмної архітектури;
3) налаштування робочого середовища та підготовка інструментів;

б) розробка back-end частини проєкту:

1) реалізація логіки міні-гри Brain Knights:

– створення статичного файлу з налаштуванням та контентом;

– створення, підключення та налаштування констант, інтерфейсів, об’єктів передачі даних, перерахованих типів, а також файлів модуля, сервісу, класу та шлюзу;

– реалізація функцій, що необхідні для реалізації геймплею;

– налаштування черги екранів;

2) інтеграція штучного інтелекту до міні-гри Turing Test:

– аналіз моделей GPT;

– створення та налаштування модулів і сервісів;

– інтеграція штучного інтелекту, його налаштування та обробка помилок, які можуть виникнути;

– підключення розробленого функціоналу до гри;

в) тестування та оптимізація:

1) виконання ручного тестування;

2) покращення продуктивності та виправлення виявлених недоліків.

Етапи розробки ігрового програмного застосунку ретельно сплановані, враховуючи взаємозв’язок процесів Це дозволяє ефективно використовувати ресурси та забезпечує своєчасне завершення проєкту в межах встановлених термінів. Також для розроблюваного ігрового програмного застосунку у жанрі multiplayer party games було створено специфікацію (див. додаток В), що містить загальний опис проєкту та його конкретні вимоги. Це допомагає забезпечити однозначне розуміння вимог до системи та сприяє виконанню завдань з високою точністю та ефективністю на всіх етапах розробки.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

Для розроблюваного ігрового програмного застосунку у жанрі multiplayer party games було створено діаграму розгортання (див. рис. 3.1) [6].

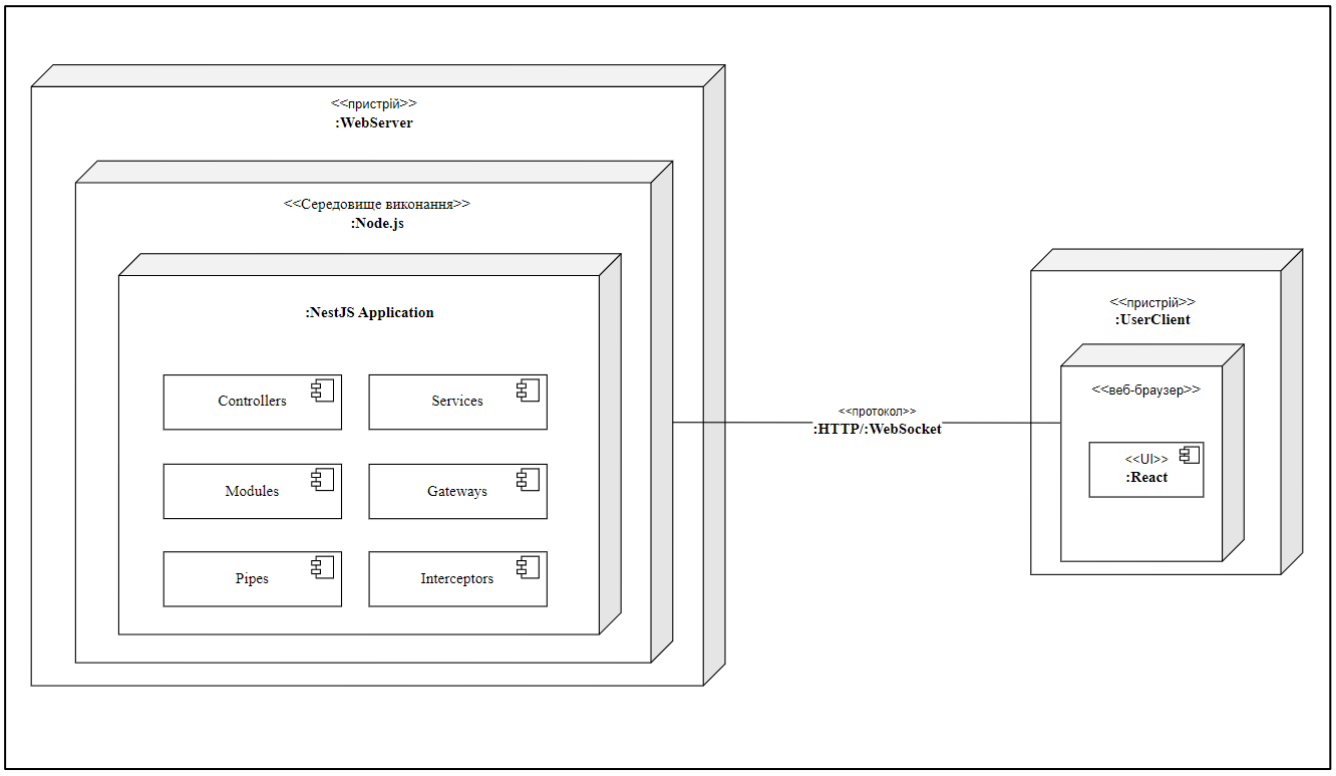


Рисунок 3.1 – UML діаграма розгортання ігрового програмного застосунку (діаграма виконана самостійно)

Ігровий програмний застосунок складається з двох компонентів: веб-сервера і веб-застосунку. Веб-сервер відповідає за стабільну роботу мережі, маршрутизацію запитів, налаштування шлюзів веб-сокетів, обробку, управління та збереження даних ігрової сесії. Веб-застосунок забезпечує зручний та інтуїтивно зрозумілий інтерфейс для взаємодії з системою через веб-браузер, комфортне керування, відображення інформації, передачу даних на сервер та обробку отриманих даних.

Для розроблюваної міні-гри Brain Knights на back-end частині ігрового програмного застосунку у жанрі multiplayer party games було створено діаграму прецедентів (див. рис. 3.2), діаграму пакетів (див. рис. 3.3) та діаграму взаємодії (див. рис. 3.4).

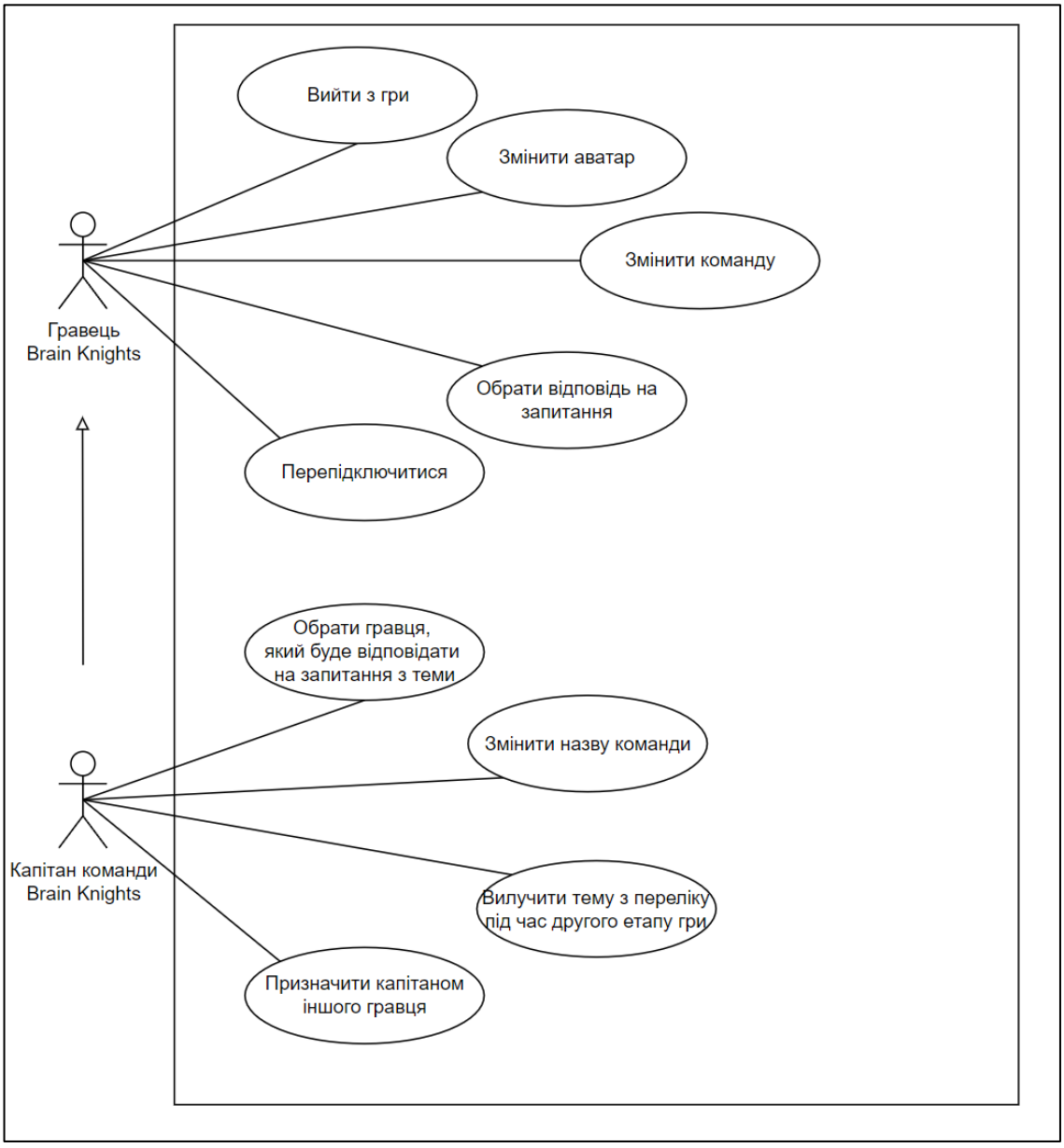


Рисунок 3.2 – UML діаграма прецедентів міні-гри Brain Knights на back-end частині ігрового програмного застосунку (діаграма виконана самостійно)

Гравець може взаємодіяти з системою наступним чином:

– вийти з гри;

– змінити аватар;

– змінити команду;

– обрати відповідь на запитання;

– перепідключитися.

Капітан може взаємодіяти з системою наступним чином:

– вийти з гри;

– змінити аватар;

– змінити команду;

– обрати відповідь на запитання;

– перепідключитися;

– обрати гравця, який буде відповідати на запитання з теми;

– змінити назву команди;

– вилучити тему з переліку під час другого етапу гри;

– призначити капітаном іншого гравця.

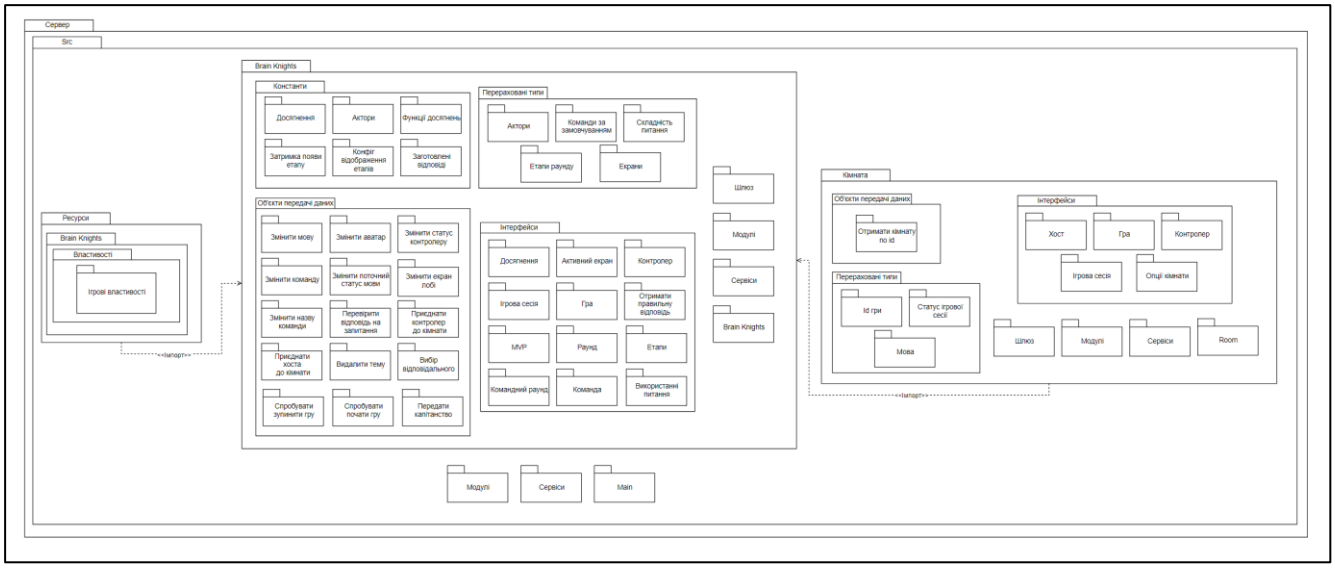


Рисунок 3.3 – UML діаграма пакетів міні-гри Brain Knights на back-end частині ігрового програмного застосунку (діаграма виконана самостійно)

Діаграма пакетів міні-гри Brain Knights на back-end частині відображає організацію та розташування елементів моделі проєкту. На ній представлена структура даних програми, а також відображені створені та налагоджені залежності між компонентами.

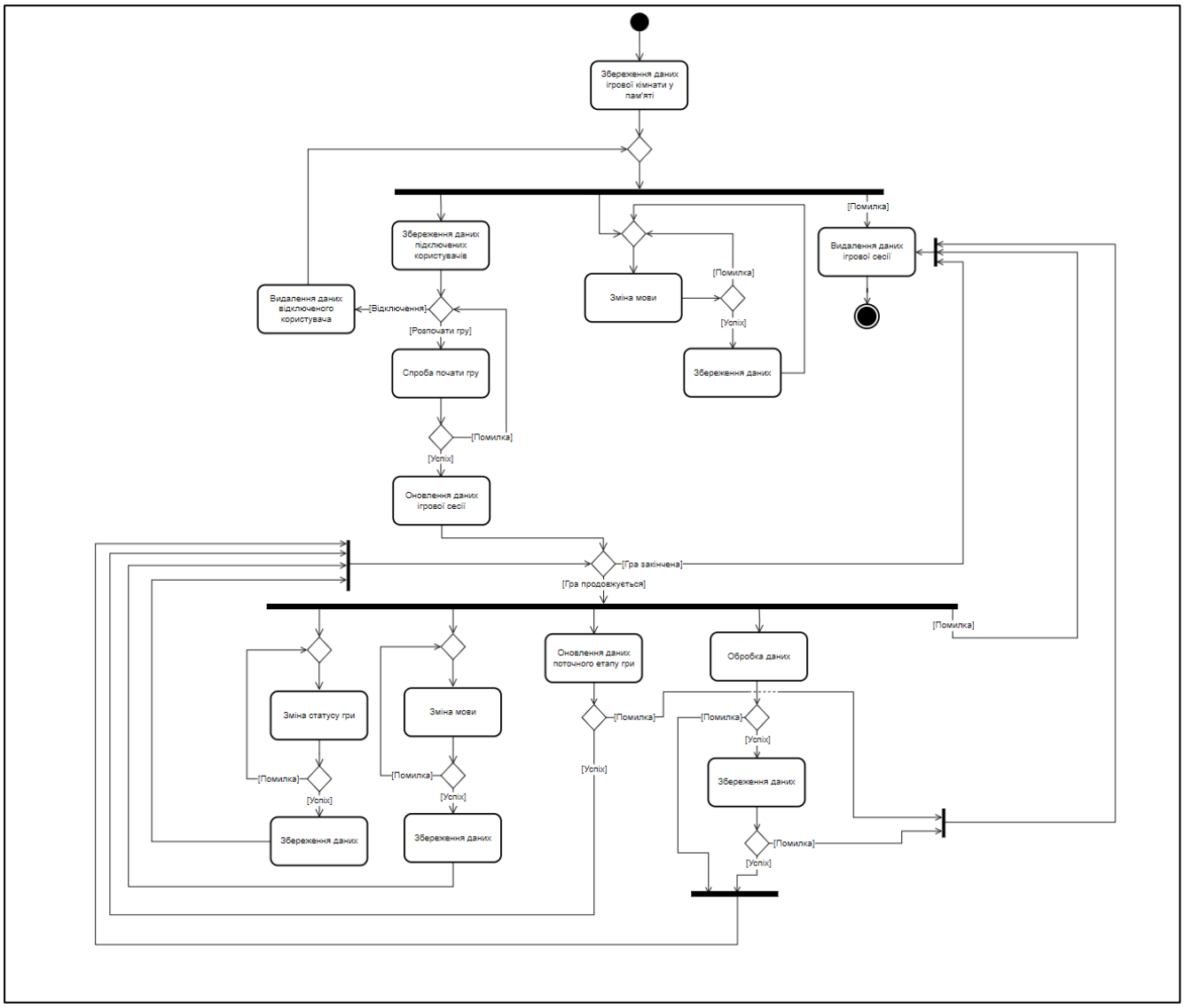


Рисунок 3.4 – UML діаграма діяльності міні-гри Brain Knights на back-end частині ігрового програмного застосунку (діаграма виконана самостійно)

Діаграма діяльності міні-гри Brain Knights на back-end частині відображає динамічні аспекти поведінки системи. На ній демонструється послідовна реалізація логіки системи, як потік керування, що переходить від однієї дії до іншої.

Для розроблюваної інтеграції штучного інтелекту на back-end частині ігрового програмного застосунку у жанрі multiplayer party games було створено діаграму прецедентів (див. рис. 3.5), діаграму компонентів (див. рис. 3.6), діаграму взаємодії (див. рис. 3.7), діаграму діяльності (див. рис. 3.8), діаграму пакетів (див. рис. 3.9) та діаграму станів (див. рис. 3.10).

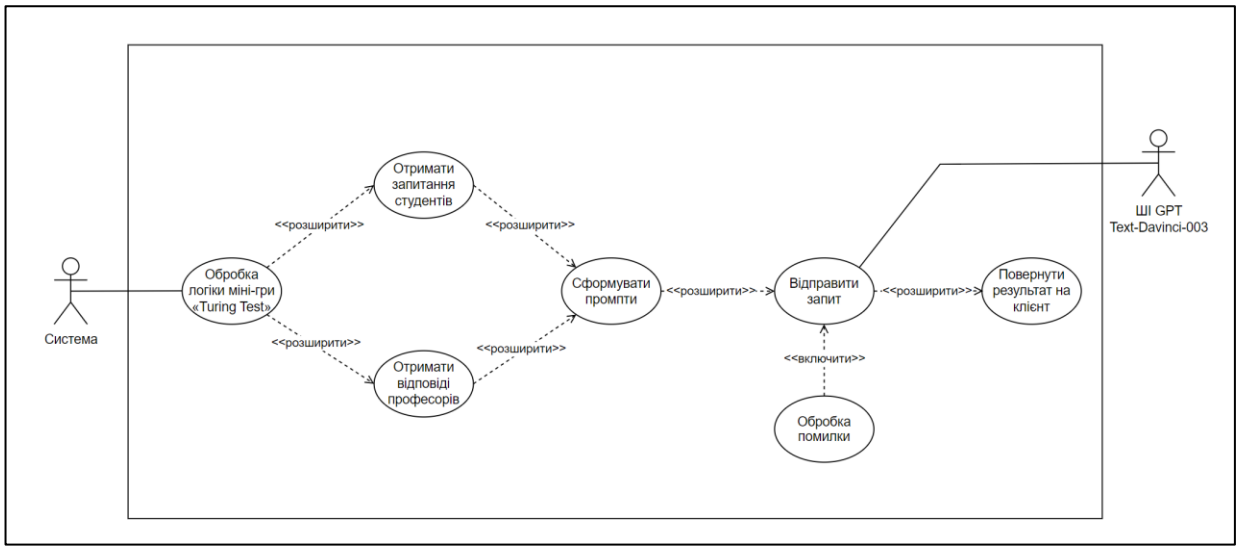


Рисунок 3.5 – UML діаграма прецедентів інтеграції ШІ на back-end частині ігрового програмного застосунку (діаграма виконана самостійно)

Система здійснює наступні дії:

– обробка логіки міні-гри Turing Test;

– розподіл отриманих даних ігрової сесії на запитання студентів та відповіді професорів;

– формування промптів;

– відправлення запиту через API компанії OpenAI на базі моделі GPT Text-Davinci-003;

– обробка можливих помилок;

– повернення результату на клієнтську частину.

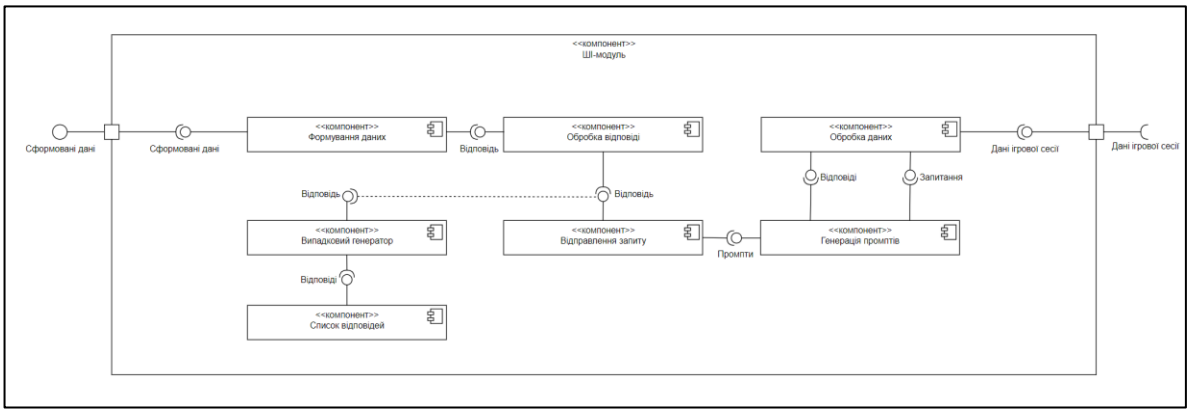


Рисунок 3.6 – UML діаграма компонентів інтеграції ШІ на back-end частині ігрового програмного застосунку (діаграма виконана самостійно)

Діаграма компонентів інтеграції ШІ на back-end частині відображає взаємозв’язок між різними компонентами ігрового програмного застосунку, що відповідають за обробку логіки міні-гри Turing Test під час етапу відповіді бота. Ці компоненти включають у себе обробку отриманих даних, генерацію промптів, відправлення запиту, обробку помилок та повернення оброблених і сформованих даних на клієнтську частину системи.

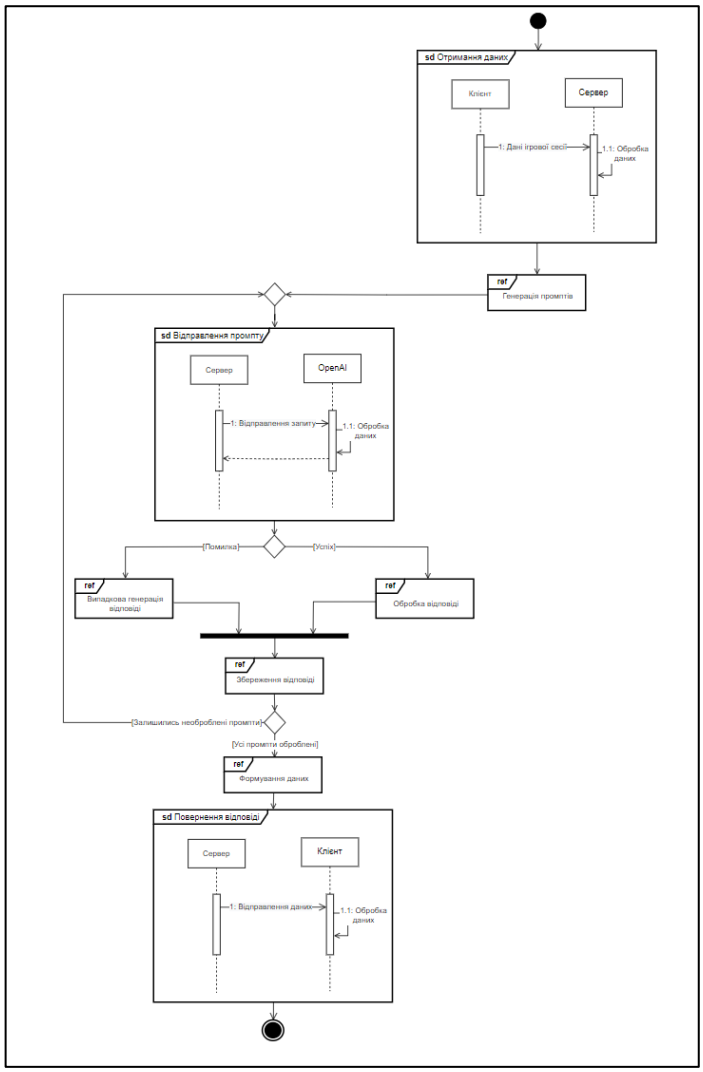


Рисунок 3.7 – UML діаграма взаємодії інтеграції ШІ на back-end частині ігрового програмного застосунку (діаграма виконана самостійно)

Діаграма взаємодії інтеграції ШІ на back-end частині зосереджується на описі потоку повідомлень всередині системи. Ця діаграма відображає послідовність логічних кроків, які виконує розроблений застосунок, та взаємодію між

компонентами системи, показуючи, як дані обробляються та передаються всередині програми.

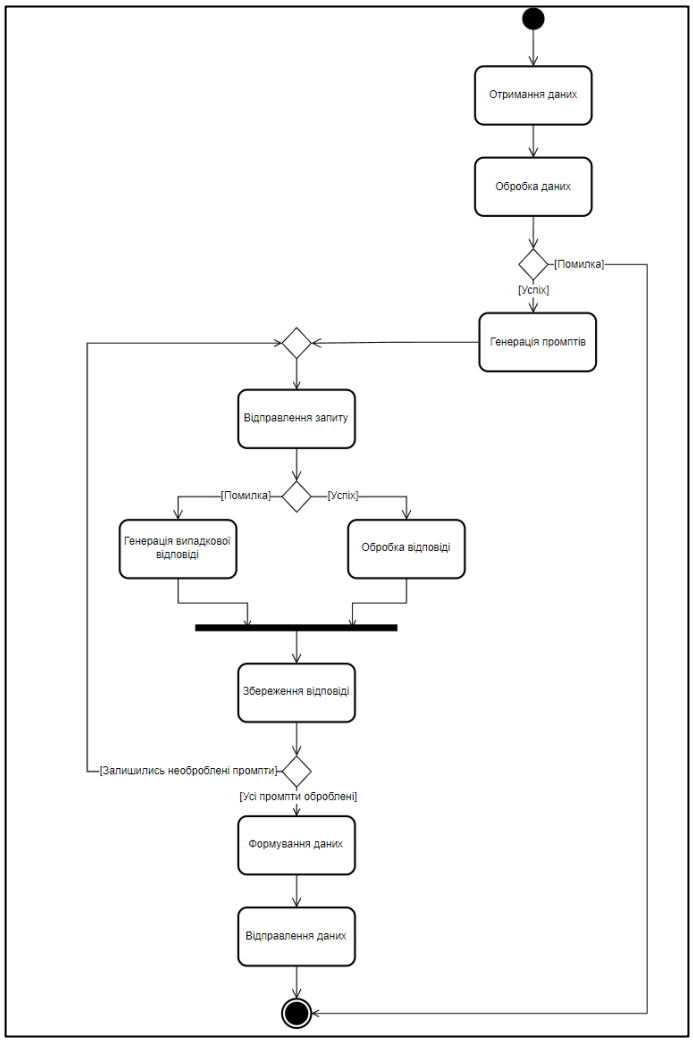


Рисунок 3.8 – UML діаграма діяльності інтеграції ШІ на back-end частині ігрового програмного застосунку (діаграма виконана самостійно)

Діаграма діяльності інтеграції ШІ на back-end частині відображає динамічні аспекти поведінки системи. Вона ілюструє послідовну реалізацію логіки системи, як потік керування, що переходить від однієї дії до іншої, створюючи можливість візуального спостереження за взаємодією компонентів системи під час інтеграції штучного інтелекту.

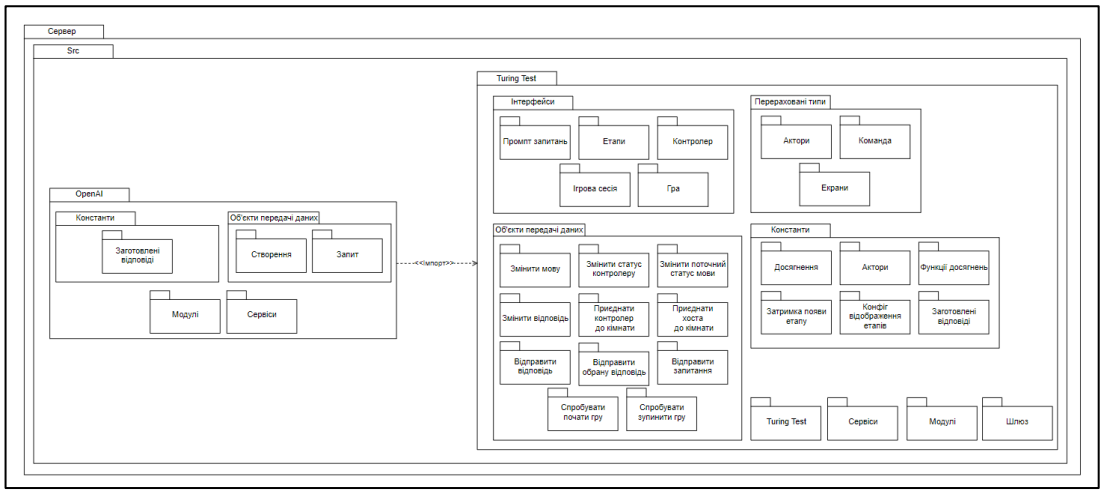


Рисунок 3.9 – UML діаграма пакетів інтеграції ШІ на back-end частині ігрового програмного застосунку (діаграма виконана самостійно)

На діаграмі пакетів інтеграції ШІ на back-end частині зображено організацію та розміщення елементів моделі проєкту. Вона також відображає структуру даних програми та залежності між компонентами, які були створені та налагоджені.

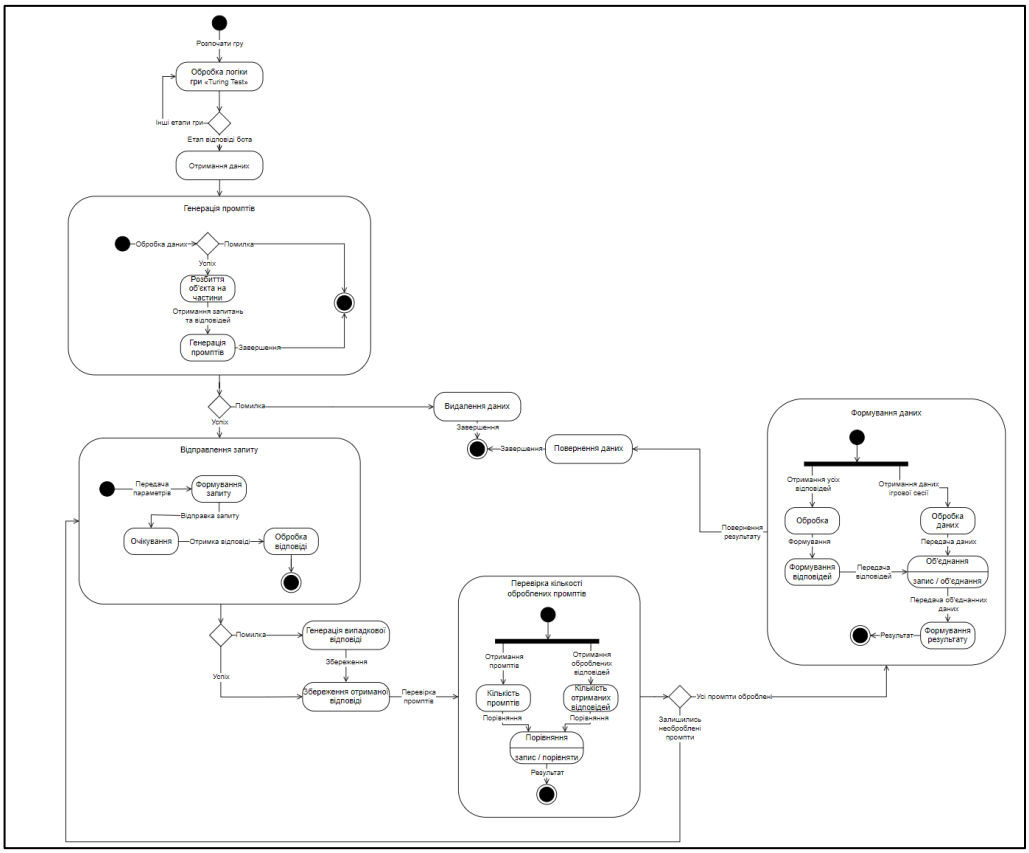


Рисунок 3.10 – UML діаграма станів інтеграції ШІ на back-end частині ігрового програмного застосунку (діаграма виконана самостійно)

Діаграма станів інтеграції ШІ на back-end частині демонструє, як об’єкти переходять з одного стану до іншого. Ця діаграма описує процес обробки станів міні-гри Turing Test під час етапу відповіді бота на запитання. Цей процес включає в себе отримання та обробку даних, генерацію промптів, відправлення запитів, валідацію помилок, формування даних та їх повернення на клієнську частину проєкту.

3.2 Проєктування архітектури ПЗ

В якості архітектури, на якій базується back-end частина програмної системи, обрана трьохрівнева архітектура (див. рис. 3.11), що дозволяє легко масштабувати проєкт, розподіляти навантаження, а також забезпечувати модульність та гнучкість системи [7]. Вона складається з трьох рівнів: представлення, бізнес-логіки та доступу до даних.

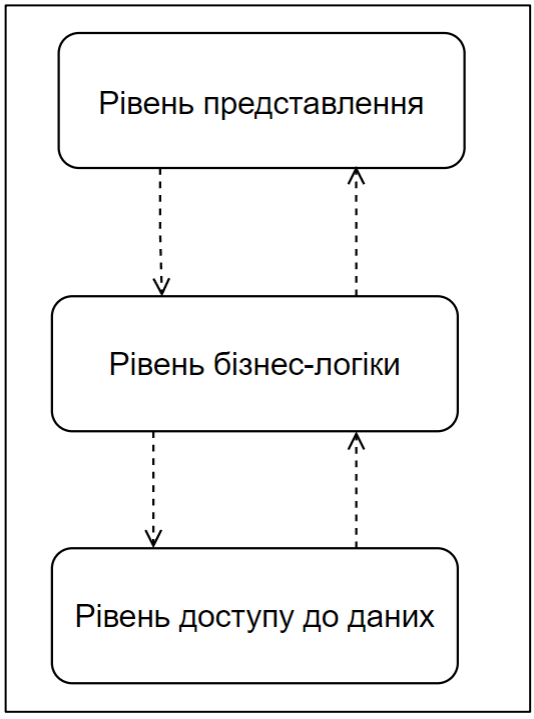


Рисунок 3.11 – Архітектура back-end частини проєкту (діаграма виконана самостійно)

Рівень представлення відповідає за взаємодію та формування даних для користувача, що взаємодіє з системою. На цьому рівні розміщуються контролери та шлюзи, які реалізовані за допомогою фреймворку NestJS на програмній платформі Node.js. Контролери відповідають за обробку HTTP-запитів, що надходять від клієнта, та формування відповіді, яка буде надана сервером. Шлюзи використовуються для обробки WebSocket-запитів, що забезпечують двонаправлену комунікацію в реальному часі, дозволяючи обмінюватися даними без необхідності постійного відправлення запитів.

Рівень бізнес-логіки відповідає за обробку даних та виконання бізнес-процесів програми. На цьому рівні розміщуються сервіси, які виконують операції, пов’язані з обробкою даних відповідно до встановлених бізнес-правил.

Рівень доступу до даних відповідає за зберігання та управління даними. Він складається з сервісів та класів, які надають доступ до цих даних.

Таким чином, трирівнева архітектура забезпечує чітке розподілення функцій між різними частинами системи, що сприяє підвищенню ефективності та гнучкості розробки і підтримки програмного забезпечення.

Для back-end частини системи було спроєктовано наступну структуру (див. рис. 3.12):

– папка assets містить статичні ресурси розроблених ігор, зокрема: їх налаштування, підв’язані аватари та меми, створена база запитань та відповідей для міні-гри Brain Knights, а також ситуації для міні-гри Skibidy Party;

– папка brain-knights містить файли, що відповідають за створену та налаштовану логіку міні-гри Brain Knights;

– папка config містить файли, пов’язані з конфігурацією застосунку;

– папка file містить файли, що відповідають за управління файлами, що завантажуються для створення власного пакету мемів у міні-грі Skibidy Party;

– папка openai містить файли, що відповідають за налаштування відправки запитів до API OpenAI;

– папка request-loger містить файли, що забезпечують логування отриманих запитів від клієнтської частини;

– папка room містить файли, що відповідають за налаштування базової логіки роботи з кімнатами;

– папка skibidy-party містить файли, що відповідають за створену та налаштовану логіку міні-гри Skibidy Party;

– папка turing-test містить файли, що відповідають за створену та налаштовану логіку міні-гри Turing Test;

– папка utils містить допоміжні файли та функції.

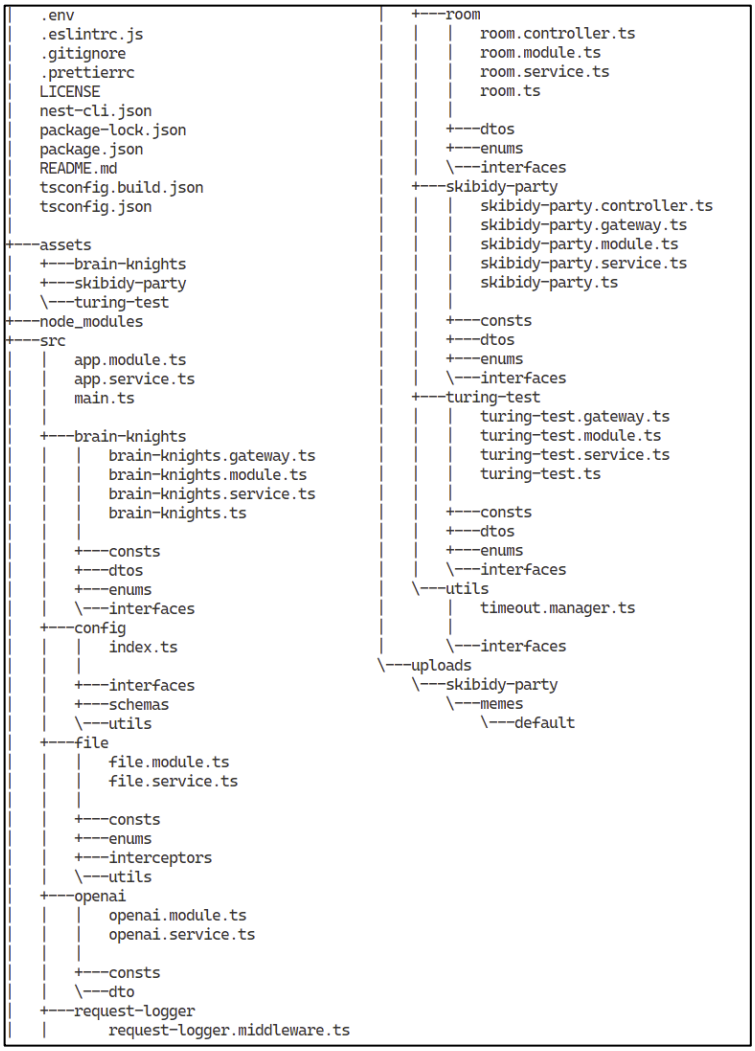


Рисунок 3.12 – Структура back-end частини проєкту (знімок екрана виконано самостійно)

Ця структура back-end частини дозволяє розділити код на окремі, взаємопов’язані компоненти, що сприяє структуруванню проєкту, забезпечує його подальшу модернізацію та підтримку у майбутньому.

3.3 Приклади найцікавіших алгоритмів та методів

Для інтеграції штучного інтелекту на серверній частині ігрового програмного додатку у жанрі multiplayer party games необхідно створити клас ChatGptService для відправки запитів до API OpenAI. Крім того, необхідно реалізувати методи generateQuestionsAnswersPrompt і addChatGptAnswer для формування промптів і даних відповіді відповідно [8-9].

Метод generateQuestionsAnswersPrompt отримує поточні дані ігрової сесії, які валідуються та розбиваються на окремі фрагменти: запитання від команди студентів та відповіді від команди професорів. Потім, через цикл, що дорівнює кількості отриманих запитань, формуються промпти, які додаються до масиву для використання при відправленні запитів. Формат промпта наступний: «Here’s my question: ‘${запитання_команди_студентів}’, and here are people’s answers to it: ‘${відповіді_команди_професорів}’. Your task is to disguise yourself as a human, and to do this you need to generate an answer similar to other people’s answers. If all other people answered incorrectly, you answer incorrectly, but similarly. If at least one person answered correctly, you answer correctly, but similarly. The number of characters in your answer should not be more than 2 times the average number of characters among other answers» [10].

У класі ChatGptService створюється метод для передачі промптів через цикл. За допомогою бібліотеки Axios виконується запит POST до API OpenAI. У тілі запиту передаються назва використовуваної моделі (у нашому випадку GPT Text-Davinci-003), промпт, температура для контролю передбачуваності відповіді та максимальна кількість токенів для обмеження довжини відповіді [11]. У заголовках передається API-ключ, який знаходиться у файлі конфігурації .env.

Після отримання відповіді потрібно її обробити. Якщо виникає будь-яка помилка, потрібно випадково повернути одну з загальновживаних відповідей, які містяться в попередньо підготованому списку на сервері.

Останнім кроком є обробка отриманої відповіді та повернення сформованих даних на клієнтську частину за допомогою функції addChatGptAnswer для наступного етапу гри – вибір відповіді бота серед отриманих варіантів відповідей. Таким чином, описана логіка інтеграції штучного інтелекту у серверну частину додатку забезпечить швидкість роботи та обробки дій гравців, валідацію помилок, а також захоплюючий ігровий досвід.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

Back-end частина ігрового програмного застосунку у жанрі multiplayer party games була розроблена за допомогою мови програмування TypeScript, програмної платформи Node.js та фреймворку NestJS. Також, були використані наступні бібліотеки:

– @nestjs/common, що використовується для основних загальних модулів і утиліт при створенні додатків на NestJS;

– @nestjs/config, що використовується для управління конфігурацією додатку, завантажуючи змінні середовища та конфігураційні файли;

– @nestjs/core, що використовується як основний модуль ядра NestJS, який забезпечує функціональність додатку;

– @nestjs/platform-express, що використовується для інтеграції NestJS з Express, популярним HTTP-сервером для Node.js;

– @nestjs/platform-socket.io, що використовується для інтеграції NestJS з Socket.IO, бібліотекою для роботи з веб-сокетами;

– @nestjs/serve-static, що використовується для сервірування статичних файлів, таких як HTML, CSS, JavaScript і зображення;

– @nestjs/websockets, що використовується для роботи з веб-сокетами в NestJS, дозволяючи створювати додатки з двостороннім зв’язком;

– @types/uuid, що використовується для надання типів TypeScript для бібліотеки uuid, яка генерує унікальні ідентифікатори (UUID);

– axios, що використовується для виконання HTTP-запитів, підтримуючи проміси та асинхронні операції;

– class-transformer, що використовується для перетворення об’єктів між різними типами і структурами у поєднанні з class-validator;

– class-validator, що використовується для валідації об’єктів на основі декораторів у поєднанні з class-transformer;

– cookie-parser, що використовується як middleware для розбору файлів cookie, полегшуючи роботу з cookie у додатках Express;

– crypto, що використовується для виконання криптографічних операцій, таких як хешування, шифрування і створення цифрових підписів;

– helmet, що використовується як middleware для забезпечення безпеки додатків Express, додаючи різні HTTP-заголовки для захисту від деяких уразливостей;

– joi, що використовується для валідації даних, дозволяючи визначити схему валідації та перевіряти дані на відповідність цій схемі;

– socket.io, що використовується для роботи з веб-сокетами, забезпечуючи двосторонній зв’язок між клієнтом і сервером у реальному часі;

– uuid, що використовується для генерації унікальних ідентифікаторів (UUID).

Для налаштування змінних середовища та зберігання конфіденційних налаштувань було створено файл .env, що містить наступні змінні, які використовуються для функціонування всієї back-end частини системи, а також для налаштування логіки міні-гри Brain Knights та інтеграції штучного інтелекту до міні-гри Turing Test:

– змінна NODE_ENV визначає поточне середовище виконання додатка;

– змінна PORT вказує порт, на якому буде запущений сервер;

– змінна PROTOCOL вказує протокол з’єднання між клієнтом та сервером;

– змінна DOMAIN вказує доменне ім’я, на якому працює клієнтська частина системи;

– змінна CHATGPT_API_KEY вказує приватний ключ, що використовується для автентифікації та авторизації доступу до API OpenAI;

– змінна BRAIN_KNIGHTS_PROPERTIES_PATH вказує шлях до файлу з налаштуваннями та контентом для міні-гри Brain Knights.

Файл з конфігураціями міні-гри Brain Knights (див. рис. 4.1) містить наступні параметри налаштування: ідентифікатор гри, назва гри, мінімальна кількість гравців, максимальна кількість гравців, аватари гри.

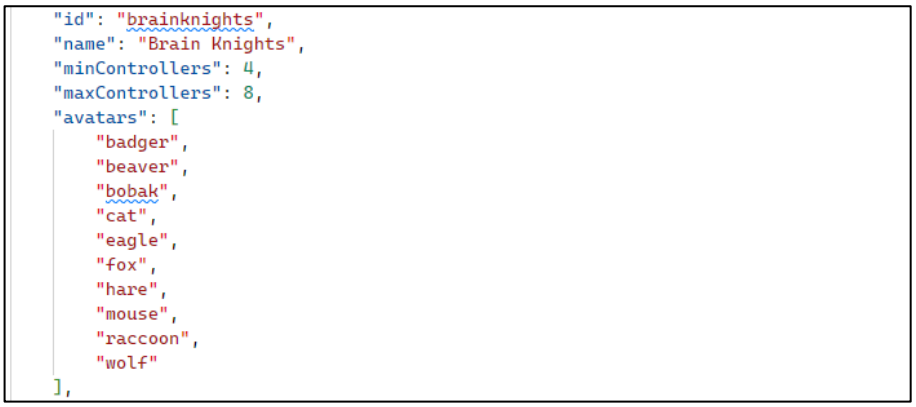


Рисунок 4.1 – Файл конфігурацій з налаштуванням міні-гри Brain Knights (знімок екрана виконано самостійно)

Контент файлу конфігурації міні-гри Brain Knights (див. рис. 4.2) має наступну структуру: масив тем, що містить ідентифікатор теми, назву теми двома мовами, масив запитань, що містить ідентифікатор запитання, саме запитання двома мовами, складність запитання та масив варіантів відповідей, що містить ідентифікатор варіанту відповіді, варіант відповіді двома мовами та прапорець, що вказує на її правильність.

Рисунок 4.2 – Файл конфігурацій з контентом міні-гри Brain Knights (знімок екрана виконано самостійно)

Для валідації конфігураційного файлу було розроблено функцію validateGameProperties, що отримує два параметри: властивості гри та схему валідації. Якщо в результаті валідації виявляється помилка, функція викидає виняток з повідомленням про помилку:

```
const validateGameProperties = (
    properties: object,
    schema: Joi.Schema,
) => {
    const result = schema.validate(properties);
    if (result.error) {
        throw new Error(result.error.message);
    }
};
```

Перед запуском back-end системи, файл конфігурацій перевіряється на відповідність за допомогою функції validateGameProperties, якій передається вміст файлу та схема його структури:

```
const brainKnightsPropertiesPath = join(__dirname, '..', '..', process.env.BRAIN_KNIGHTS_PROPERTIES_PATH);
const [brainKnightsProperties] = [fs.readFileSync(brainKnightsPropertiesPath, 'utf-8')];
const parsedBrainKnightsProperties: BrainKnightsGame = JSON.parse(brainKnightsProperties);
validateGameProperties(
    parsedBrainKnightsProperties,
    brainKnightsPropertiesValidationSchema
);
```

Після успішної перевірки система продовжує свою роботу, а функція конфігураційного файлу повертає об’єкт, який містить змінні з файлу .env:

```
testing: process.env.NODE_ENV === 'development',
protocol: process.env.PROTOCOL,
port: parseInt(process.env.PORT, 10),
domain: process.env.DOMAIN,
chatGptApiKey: process.env.CHATGPT_API_KEY,
```

А також конфігураційні налаштування та контент для інших міні-ігор, зокрема Brain Knights:

```
brainKnights: {
    id: parsedBrainKnightsProperties.id,
    name: parsedBrainKnightsProperties.name,
    minControllers: parsedBrainKnightsProperties.minControllers,
    maxControllers: parsedBrainKnightsProperties.maxControllers,
    avatars: parsedBrainKnightsProperties.avatars,
    topics: parsedBrainKnightsProperties.topics,
}
```

Для інтеграції розробленої логіки міні-гри Brain Knights у загальну структуру сервера був створений файл модуля. У цьому файлі проводиться імпорт модулів кімнат, перераховуються використовувані сервіси – файли шлюзу та сервісів, та експортується функціональність сервісів для доступу до неї з інших модулів:

```
@Module({
    imports: [RoomModule],
    providers: [BrainKnightsGateway, BrainknightsRoomService],
    exports: [BrainknightsRoomService],
})
```

Клас BrainKnightsRoom створено для моделювання кімнати у міні-грі Brain Knights. Ця кімната взаємодіє з ігровою сесією BrainKnightsGameSession та контролером BrainKnightsController. Усередині класу є кілька приватних і публічних змінних і властивостей: timeoutManager керує часовими затримками та їх обробкою; usedTopics та usedQuestions зберігають списки використаних тем і питань відповідно; публічні змінні elapsedTime, timeSinceLastAnswer і timeSinceLastQuestionChange відстежують пройдений час з різних подій гри; questionChangeInterval визначає інтервал часу між зміною питань; secondRoundTimeSkip встановлює часову затримку для переходу на другий етап гри; correctAnswerShown вказує, чи було показано правильну відповідь. Конструктор класу ініціалізує TimeoutManager та приймає параметри для налаштування кімнати:

```
    export class BrainKnightsRoom extends Room<BrainKnightsGameSession,
BrainKnightsController> {
        private readonly timeoutManager: TimeoutManager;
        private usedTopics = [];
        private usedQuestions: UsedQuestions = {};
        public elapsedTime: number = 0;
        public timeSinceLastAnswer: number = 0;
        public timeSinceLastQuestionChange: number = 0;
        public                  questionChangeInterval                  =
SCREENS_APPEARANCE_DELAY.questionChange.timeToChangeScreen;
        public          secondRoundTimeSkip:          number          =
SCREENS_APPEARANCE_DELAY.stage12.timeToChangeScreen;
        public correctAnswerShown: boolean = false;
        constructor(options:          RoomOptions<BrainKnightsGameSession,
BrainKnightsController>,) {
            super(options);
            this.timeoutManager = new TimeoutManager();
        }
        ...
    }
```

Клас BrainknightsRoomService є сервісом, який виконує операції, пов'язані з обробкою даних відповідно до встановлених бізнес-правил у міні-грі Brain Knights. При створенні екземпляру цього класу відбувається ініціалізація приватної змінної brainKnightsProperties, яка представляє собою об'єкт типу BrainKnightsGame. Цей об'єкт отримується через ін'єкцію залежностей configService, що надає доступ до конфігурації додатка, та roomService, який дозволяє взаємодіяти з кімнатами у грі. Конструктор отримує ConfigService та RoomService як параметри, які визначаються як приватні члени класу. У конструкторі також викликається метод get з configService, щоб отримати об'єкт типу BrainKnightsGame з конфігурації гри і призначити його змінній brainKnightsProperties:

```
@Injectable()
export class BrainknightsRoomService {
    private readonly brainKnightsProperties: BrainKnightsGame;

    constructor(private readonly configService: ConfigService, private
readonly roomService: RoomService) {
        this.brainKnightsProperties                                    =
this.configService.get<BrainKnightsGame>('brainKnights');
    }
    ...
}
```

Клас BrainKnightsGateway є шлюзом WebSocket для взаємодії з клієнтами в міні-грі Brain Knights. Використані декоратори @WebSocketGateway та @UsePipes встановлюють налаштування шлюзу WebSocket, зокрема простір імен та використання об'єкта ValidationPipe для валідації даних, що отримуються. У конструкторі класу ініціалізується приватна змінна loggerService для логування подій, а також зчитується конфігурація гри через сервіс configService та призначається змінній brainKnightsProperties:

```
@WebSocketGateway(
    {
        namespace: '/brainknights',
        cors: {origin: '*'
    },
})
@UsePipes(new ValidationPipe())
export class BrainKnightsGateway implements OnGatewayConnection,
OnGatewayDisconnect
{
    private readonly loggerService: LoggerService;
    private readonly brainKnightsProperties: BrainKnightsGame;
    private intervalIdsMap: Record<string, NodeJS.Timeout[]> = {};
```

```
        constructor(private readonly brainknightsRoomService: BrainknightsRoomService, private readonly configService: ConfigService, private readonly roomService: RoomService,) {
            this.loggerService = new Logger(BrainKnightsGateway.name);
            this.brainKnightsProperties =
                this.configService.get<BrainKnightsGame>('brainKnights');
        }
        @WebSocketServer() server: Server = new Server<ServerToClientEvents, ClientToServerEvents>();
        ...
    }
```

Створений метод startFirstRoundRandomQuestionGenerationInterval в файлі шлюзів використовується для початку генерації випадкових питань першого раунду. Він встановлює інтервал, протягом якого генеруються питання, і перевіряє, чи не закінчився час раунду. Якщо час ще не вичерпано, метод збільшує загальний час та час з останньої зміни питання. Якщо час з останньої зміни питання досягнув інтервалу зміни питань, вибирається випадкове питання для кімнати, і інформація про оновлення кімнати відправляється на клієнт. Після завершення часу раунду метод визначає переможця раунду, оновлює дані кімнати та очищає інтервал:

```
    private startFirstRoundRandomQuestionGenerationInterval(room: BrainKnightsRoom, duration: number) {
        const roomId = room.getId();
        const intervalId = setInterval(() => {
            if (room.elapsedTime < duration) {
                if (!room.timeoutsIsPaused()) {
                    room.elapsedTime += SCREENS_APPEARANCE_DELAY.interval.timeToChangeScreen;
                    room.timeSinceLastQuestionChange += SCREENS_APPEARANCE_DELAY.interval.timeToChangeScreen;
                }
                if (room.timeSinceLastQuestionChange >= room.questionChangeInterval && room.elapsedTime < duration - SCREENS_APPEARANCE_DELAY.shorterTimeToGenerateQuestions.timeToChangeScreen) {
    this.brainknightsRoomService.chooseRandomQuestion(roomId);
                    this.server.to(roomId).emit('roomUpdated', { room });
                    this.loggerService.log(`Random question chosen in room ${roomId}`);
                    room.questionChangeInterval = SCREENS_APPEARANCE_DELAY.questionChange.timeToChangeScreen;
                    room.timeSinceLastQuestionChange = 0;
                }
            } else if (room.elapsedTime >= duration) {
                const controllerFirstSubRoundWinner = this.brainknightsRoomService.getFirstSubRoundWinnerAndClearData(roomId);
                if (controllerFirstSubRoundWinner) {
    this.server.to(roomId).emit('controllerBrainKnightsFirstSubRoundWinner', {controllerFirstSubRoundWinner});
                }
```

```
                room.elapsedTime = 0;
                room.timeSinceLastQuestionChange = 0;
                room.questionChangeInterval =
SCREENS_APPEARANCE_DELAY.questionChange.timeToChangeScreen;
                clearInterval(intervalId);
            }
        }, SCREENS_APPEARANCE_DELAY.interval.timeToChangeScreen);
        this.addToIntervalIdsMap(roomId, intervalId);
    }
```

Метод startSecondRoundRandomQuestionGenerationInterval в цілому схожий з методом startFirstRoundRandomQuestionGenerationInterval і використовується для генерації випадкових питань у другому раунді гри. Однак другий метод має додаткову умову, яка перевіряє, чи настав час показу правильної відповіді на питання у другому раунді. Якщо час між останньою відповіддю та показом відповіді потрапляє в певний діапазон, метод виконує додаткові дії, такі як зміна затримки перед показом відповіді та відправка правильної відповіді клієнтам:

```
    if (room.timeSinceLastAnswer >=
SCREENS_APPEARANCE_DELAY.secondRoundAnswerOnOneQuestion.timeToChangeScreen
&& room.timeSinceLastAnswer <=
SCREENS_APPEARANCE_DELAY.secondRoundShowAnswerOnOneQuestion.timeToChangeScr
een) {
        if (!room.correctAnswerShown) {
            if (room.secondRoundTimeSkip %
SCREENS_APPEARANCE_DELAY.secondRoundShowAnswerOnOneQuestion.timeToChangeScr
een === 0) {
                room.secondRoundTimeSkip = room.secondRoundTimeSkip -
SCREENS_APPEARANCE_DELAY.secondRoundShowAnswerOnOneQuestion.timeToChangeScr
een;
            } else {
                room.secondRoundTimeSkip = room.secondRoundTimeSkip -
(SCREENS_APPEARANCE_DELAY.secondRoundShowAnswerOnOneQuestion.timeToChangeSc
reen -
SCREENS_APPEARANCE_DELAY.secondRoundAnswerOnOneQuestion.timeToChangeScreen)
;
            }
            const correctAnswerId =
this.brainknightsRoomService.getCorrectAnswer(roomId);
            const answers =
this.brainknightsRoomService.getCorrectAnswerWithDelayAndScorePoints(roomId
);
            if (correctAnswerId && answers) {
                room.correctAnswerShown = true;

this.server.to(roomId).emit('getBrainKnightsCorrectAnswer',
{correctAnswerId});

this.server.to(roomId).emit('isBrainKnightsCorrectAnswerSecondRound',
{answers});
                this.server.to(roomId).emit('roomUpdated', { room });
            }
    }}
```

Для створених етапів міні-гри було створено масив об’єктів, що містить назву етапу та масив екранів, які включають: виклик функції для створення екрану roomCreateTimeoutFunction; активний екран activeScreen; затримку перед зміною екрана delay; повідомлення для логування logMessage; назву екрану screenName; для кого призначений екран screenFor; масив функцій, які знаходяться в класі roomTimeoutCallbacks, сервісах serviceTimeoutCallbacks та шлюзах thisTimeoutCallbacks, які необхідно виконати; а також етап гри stage:

```
name: SCREENS_APPEARANCE_DELAY.stage5.screenName,
screens: [
    {
        roomCreateTimeoutFunction: 'createHostScreenTimeout',
        activeScreen: BrainKnightsScreenEnum.HOST_SUB_ROUND_RESULT,
        delay: SCREENS_APPEARANCE_DELAY.stage5.timeToChangeScreen,
        logMessage: 'host',
        screenName: { name: 'activeScreen', for: 'host' },
        screenFor: ActorsEnum.HOST,
        roomTimeoutCallbacks: ['removeAllPersonalScreens'],
        serviceTimeoutCallbacks: [],
        thisTimeoutCallbacks: [],
        stage: RoundStageEnum.FIRST_ROUND,
    },
    {
        roomCreateTimeoutFunction: 'createControllerScreenTimeout',
        activeScreen:
BrainKnightsScreenEnum.CONTROLLER_SUB_ROUND_RESULT,
        delay: SCREENS_APPEARANCE_DELAY.stage5.timeToChangeScreen,
        logMessage: 'controller',
        screenName: { name: 'activeScreen', for: 'controller' },
        screenFor: ActorsEnum.CONTROLLER,
        roomTimeoutCallbacks: [],
        serviceTimeoutCallbacks: [],
        thisTimeoutCallbacks: [],
        stage: RoundStageEnum.FIRST_ROUND,
    },
]
```

Для інтеграції сервісу OpenAI у загальну структуру сервера був створений файл модуля. У цьому файлі перераховується використовуваний сервіс та експортується його функціональність для доступу до неї з інших модулів:

```
@Module({
    providers: [ChatGptService],
    exports: [ChatGptService],
})
export class ChatGptModule {}
```

Метод generateText асинхронно генерує текст на основі наданого промпту і моделі за допомогою API OpenAI. Метод приймає об’єкт типу CreateChatgptDto,

який містить промпт і модель, і робить POST-запит до кінцевої точки API OpenAI з цими параметрами. У запиті також передаються заголовки, включаючи Content-Type та authorization, де використовується ключ API для аутентифікації. Якщо запит успішний, метод повертає отримані дані. У разі помилки, метод логує повідомлення про невдачу і повертає випадкову відповідь, згенеровану методом generateRandomAnswer:

```
    async generateText({ prompt, model }: CreateChatgptDto) {
        try {
            const response = await axios.post<ChatGptResponse>(
                'https://api.openai.com/v1/completions',
                {model, prompt, temperature: 1, max_tokens: 100},
               {
                    headers: {
                       'Content-Type': 'application/json',
                        authorization: `Bearer ${this.apiKey}`,
                    },
                },
            );
            return response.data;
        } catch (error: any) {
            const textColor = '\x1b[0m';
            const logMessage = `${textColor} Failed to generate text: ${
                error.message || 'Unknown error'
            }`;
            this.loggerService.log(logMessage);
            return   {choices:   [{   text:   this.generateRandomAnswer(),
log_probs: [] }]};
        }
    }
    private generateRandomAnswer(): string {
        return        UNIVERSAL_ANSWERS[Math.floor(Math.random()        *
UNIVERSAL_ANSWERS.length)];
    }
```

Такі програмні рішення дозволяють досягти ефективної взаємодії між компонентами системи та забезпечити високу якість користувацького досвіду.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування має охоплювати всі розроблені функції back-end частини системи в рамках кваліфікаційної роботи, що включає реалізацію логіки міні-гри Brain Knights та інтеграцію штучного інтелекту до міні-гри Turing Test.

Зважаючи на невеликий розмір проєкту, складність функціональності та налаштований зв’язок клієнтської частини з back-end частиною програмного застосунку, було вирішено використовувати ручне тестування. Цей метод забезпечує детальне опрацювання та виявлення критичних проблем, оскільки дозволяє тестувальнику активно взаємодіяти з системою та оперативно реагувати на знайдені недоліки. Ручне тестування сприяє швидкому виявленню та виправленню помилок, оскільки тестувальник може безпосередньо відслідковувати роботу системи. Тестування програмного забезпечення складалося з наступних кроків:

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

б) перевірка функціональних вимог:

1) перевірка розробленої логіки міні-гри Brain Knights:

– перевірено можливість підключення до ігрової сесії;

– перевірено можливість відключення від ігрової сесії;

– перевірено можливість перепідключення до ігрової сесії;

– перевірено, що система надає статус власника кімнати першому підключеному гравцю;

– перевірено, що система передає статус власника кімнати наступному користувачу у списку, якщо власник відключився;

– перевірено, що власник кімнати може розпочати ігрову сесію за умови підключення необхідного мінімуму гравців та підтвердження їхньої готовності;

– перевірено, що хост-користувач може ставити гру на паузу;

– перевірено, що хост-користувач може відновлювати ігровий процес;

– перевірено, що система відключає всіх інших гравців у разі відключення хост-користувача;

– перевірено, що система очищує дані ігрової сесії у разі відключення хост-користувача;

– перевірено, що система виводить код кімнати;

– перевірено, що система виводить список гравців;

– перевірено, що система виводить дані гравців;

– перевірено, що система виводить час, що залишився до зміни екрану;

– перевірено, що система валідує отримані дані від клієнтської частини;

– перевірено, що система обробляє отримані дані від клієнтської частини;

– перевірено, що система створює чергу екранів з налаштуванням ролей;

– перевірено, що система налаштовує час для екранів;

– перевірено, що система викликає функції перед зміною екрану;

– перевірено, що система викликає функції в момент відображення екрану;

– перевірено, що система автоматично очищує дані ігрової сесії після завершення гри;

– перевірено, що система автоматично надає випадковий аватар гравцю, який увійшов у кімнату, де ігрова сесія ще не розпочалася;

– перевірено, що система надає можливість гравцям змінити аватар;

– перевірено, що система надає можливість гравцям змінити команду;

– перевірено, що система розподіляє гравців по командах, де менше користувачів;

– перевірено, що система записує гравця до червоної команди, якщо їх рівна кількість або гравець перший заходить до кімнати;

– перевірено, що система надає роль капітана першим гравцям у командах;

– перевірено, що система видаляє роль капітана у гравця, який перейшов в іншу команду, де вже є капітан;

– перевірено, що система надає можливість капітану передати свою роль іншому гравцю;

– перевірено, що система надає можливість капітану змінити назву команди;

– перевірено, що система надає можливість користувачам отримувати інформацію про команди;

– перевірено, що система надає можливість користувачам отримувати інформацію про гравців у командах;

– перевірено, що система визначає кількість раундів;

– перевірено, що система випадковим чином обирає тему для кожного раунду;

– перевірено, що система надає можливість капітану обрати гравця, який буде відповідати на запитання з обраної теми;

– перевірено, що система надає можливість користувачам отримувати інформацію про тему;

– перевірено, що система надає можливість користувачам отримувати випадкове питання;

– перевірено, що система надає можливість користувачам отримувати варіанти відповіді до питання;

– перевірено, що система надає можливість отримувати інформацію про вибрані гравцями відповіді;

– перевірено, що система надає можливість отримувати інформацію про правильність відповідей;

– перевірено, що система підраховує бали для команд у кінці кожного раунду;

– перевірено, що система надає можливість отримувати інформацію про загальний рахунок балів для кожної команди;

– перевірено, що система рахує кількість правильних відповідей кожного гравця;

– перевірено, що система визначає MVP гравців гри;

– перевірено, що система надає можливість капітанам почергово обирати теми, які необхідно вилучити зі списку;

– перевірено, що система визначає команду, яка перемогла у грі;

– перевірено, що система розподіляє досягнення гравців, здобуті протягом ігрової сесії;

2) перевірка проведеної інтеграції штучного інтелекту до міні-гри Turing Test:

– перевірено, що бот надає природні відповіді на запити користувачів;

– перевірено, що запитання студентів передаються до промпту для маскування бота;

– перевірено, що відповіді професорів передаються до промпту для маскування бота;

– перевірено, що бот може адаптувати свої відповіді залежно від різних ігрових ситуацій;

– перевірено, що бот повертає випадкову загальновживану відповідь з підготовленого списку у разі виникнення помилки під час запиту до API OpenAI;

в) перевірка нефункціональних вимог:

1) перевірено, що контент міні-гри Brain Knights у статичному файлі підтримує українську мову;

2) перевірено, що контент міні-гри Brain Knights у статичному файлі підтримує англійську мову;

3) перевірено, що кожен отриманий запит від клієнтської частини логується;

4) перевірено, що налаштування промпту для отримання відповіді від ШІ коректне та відповідає очікуваним результатам.

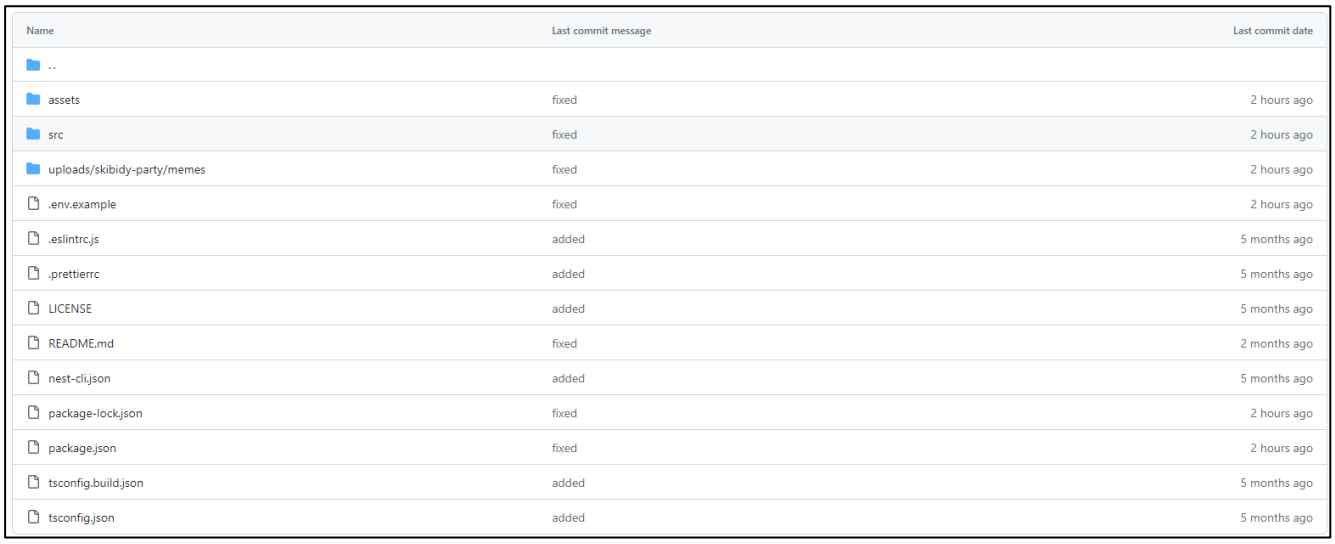
В результаті проведення тестування для деяких функціональних тестів back-end частини ігрового програмного застосунку були створені тест-кейси (див. додаток Д).

6 ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Впровадження програмного забезпечення для проєкту було здійснено шляхом розгортання на хмарній платформі Render, яка використовується для розгортання та хостингу веб-застосунків, серверів, баз даних та інших сервісів [12]. Перед розгортанням застосунку на платформі Render було проведено підготовчий етап, який включав перевірку роботи розробленої back-end частини на локальному сервері та підготовку скриптів для управління життєвим циклом проєкту у файлі package.json. Було додано новий скрипт start:prod (див. рис. 6.1), що виконує команду node dist/main для запуску системи у виробничому середовищі.

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

Наступним кроком було створено репозиторій на веб-сервісі GitHub за посиланням https://github.com/vlab1/roflsfun/tree/roflsfun/server, що містить останню версію розробленої back-end частини застосунку (див. рис. 6.2).



Name	Last commit message	Last commit date
..		
assets	fixed	2 hours ago
src	fixed	2 hours ago
uploads/skibidy-party/memes	fixed	2 hours ago
.env.example	fixed	2 hours ago
.eslintrc.js	added	5 months ago
.prettierrc	added	5 months ago
LICENSE	added	5 months ago
README.md	fixed	2 months ago
nest-cli.json	added	5 months ago
package-lock.json	fixed	2 hours ago
package.json	fixed	2 hours ago
tsconfig.build.json	added	5 months ago
tsconfig.json	added	5 months ago

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

Після цього до платформи Render було підв’язано аккаунт GitHub, створено безкоштовний веб-сервіс на Node.js та вказані команди для запуску та збірки проєкту (див. рис. 6.3).

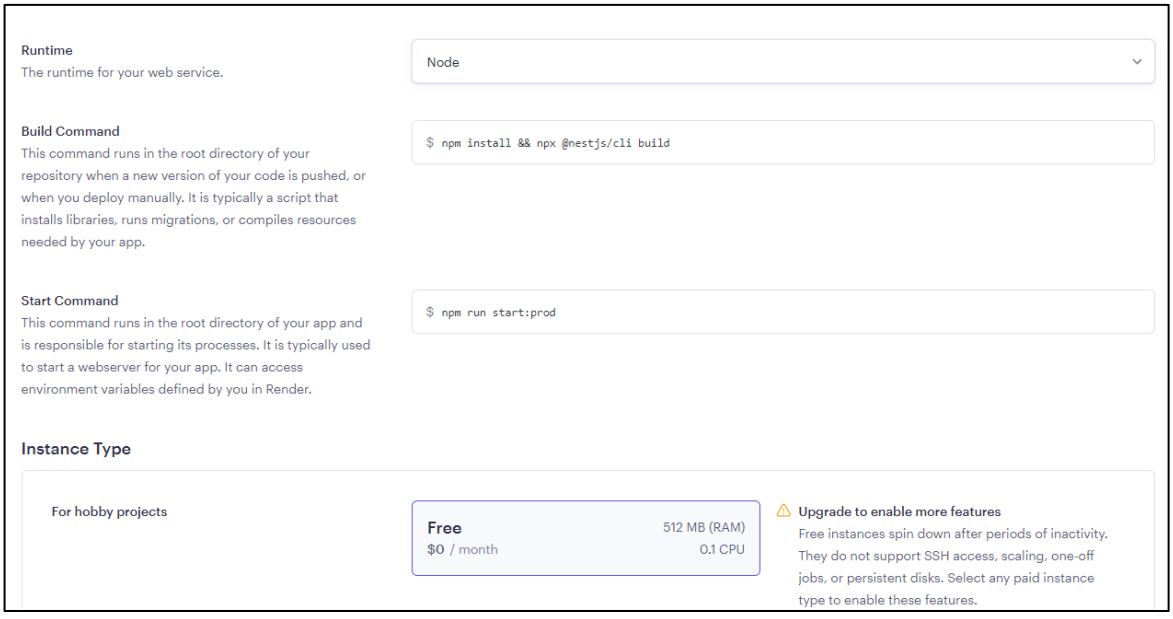


Рисунок 6.3 – Налаштування веб-сервісу на платформі Render (знімок екрана виконано самостійно)

Також були додані змінні середовища, що знаходились у файлі .env на локальному проєкті (див. рис. 6.4).

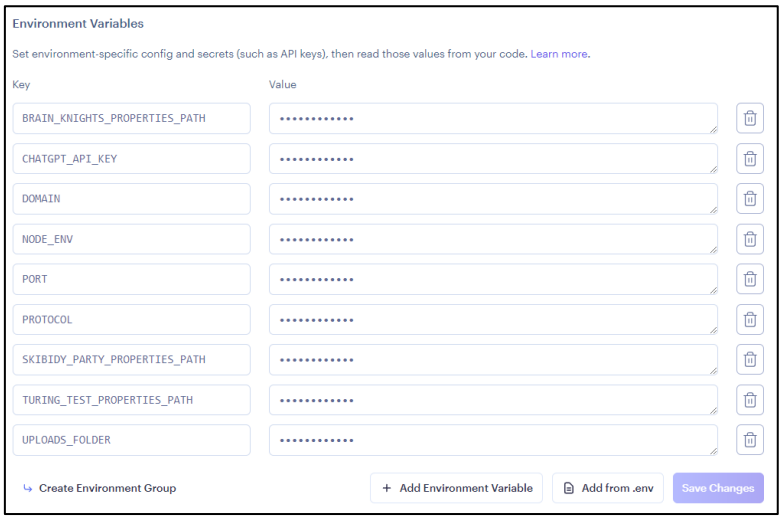


Рисунок 6.4 – Налаштування змінних середовища на платформі Render (знімок екрана виконано самостійно)

Після завершення всіх налаштувань розпочалося розгортання проєкту з підключеного GitHub-репозиторію. Потім було перевірено підключення back-end частини застосунку на клієнтській стороні системи, результат відобразився у логах (див. рис. 6.5).

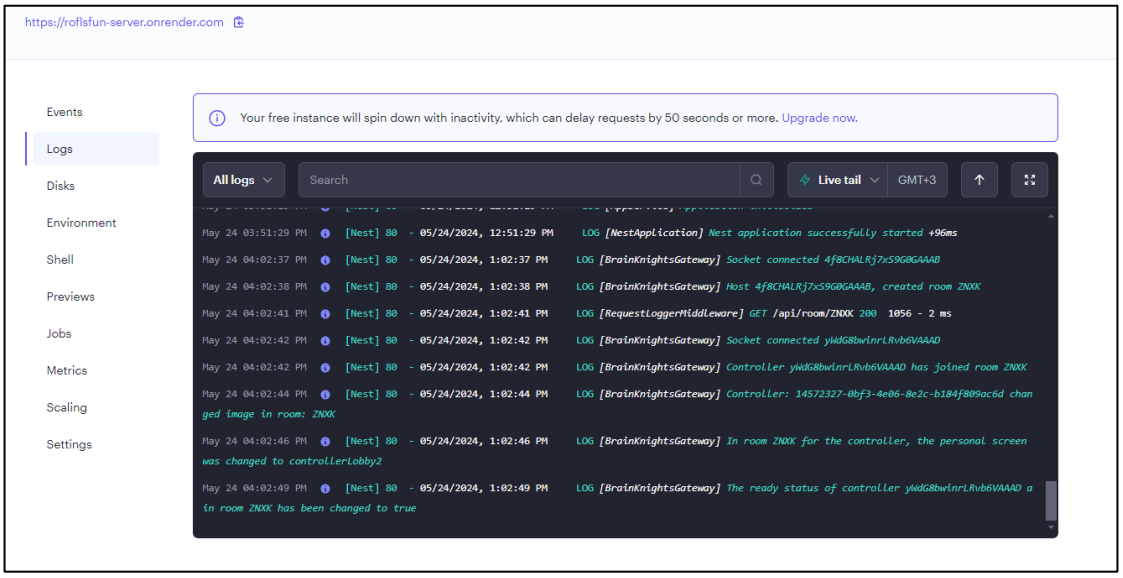


Рисунок 6.5 – Логи на розгорнутої back-end частину застосунку на платформі Render (знімок екрана виконано самостійно)

Таким чином, back-end частина ігрового програмного застосунку була успішно впроваджена в мережу та протестована.

Одним зі способів впровадження розробленого програмного забезпечення стала участь у виставці технічної творчості молоді на 28-му Міжнародному форумі «Радіоелектроніка та молодь у XXI столітті» (див. додаток Е). На цій виставці команда проєкту ROFLSUN представила ігровий програмний застосунок у жанрі multiplayer party games і отримала друге місце (див. додаток Ж).

Також було взято участь у конференції «Інформаційні інтелектуальні системи» на 28-му Міжнародному форумі «Радіоелектроніка та молодь у XXI столітті», де було представлено тези на тему «Імітація реалістичного ігрового супротивника за допомогою штучного інтелекту на основі моделі GPT Text-Davinci-003» (див. додаток И). Цю роботу було відзначено під час роботи секції №3 «Програмна інженерія. Інформаційні технології в освіті» та нагороджено грамотою за змістовну доповідь (див. додаток К).

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було успішно створено back-end частину системи та інтегровано штучний інтелект в ігровий програмний застосунок у жанрі multiplayer party games.

Back-end частина застосунку розроблена з використанням мови програмування TypeScript, платформи Node.js та фреймворку NestJS. Інтеграція штучного інтелекту до системи здійснена на back-end частині за допомогою API OpenAI моделі GPT Text-Davinci-003. Для здійснення запиту до API використано бібліотеку Axios, а для забезпечення обміну даними між клієнтською та серверною частинами в реальному часі – бібліотеку WebSocket на серверній частині проєкту.

Розроблена back-end частина системи відповідає за обробку логіки міні-гри Brain Knights, а також інтеграцію та налаштування штучного інтелекту у міні-грі Turing Test. Реалізована логіка міні-гри Brain Knights включає в себе такі базові ігрові налаштування: обмеження мінімальної та максимальної кількості гравців, створення двомовної бази тем, запитань та відповідей. Вона також забезпечує управління ігровим процесом, зокрема підключення гравців до ігрової сесії, розподіл ролей, керування станом гри, зміну та налаштування черговості екранів гри, а також обробку та валідацію отриманих даних від клієнта. Інтегрований штучний інтелект моделі GPT Text-Davinci-003 від компанії OpenAI виконує роль бота у міні-грі Turing Test, який маскується під гравців та імітує природні відповіді на основі отриманих варіантів відповідей від команди професорів та поставлених запитань від команди студентів, що покращує рівень зацікавленості та задоволення від розробленого геймплею.

Таким чином, можна стверджувати, що мета кваліфікаційної роботи була досягнута. Розроблений ігровий програмний застосунок у жанрі multiplayer party games може бути використаний у компанії друзів і знайомих, які прагнуть провести час в онлайн-іграх. Можливість грати на будь-якому комп’ютері або консолі з Інтернет-підключенням та веб-браузером як хост, а також використовувати веб-

браузер смартфону або іншого пристрою як контролер для гри робить програмний продукт максимально зручним та доступним для всіх користувачів. Крім того, можна зазначити, що він відповідає сучасним тенденціям у розвитку ігрової індустрії та може стати популярним серед широкого кола користувачів, якщо надалі продовжувати його розвиток та систематично оновлювати.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Baumgartner S. TypeScript in 50 Lessons / ред. R. Andrew. Germany : Smashing Magazine, 2020. 464 с.

2. Index | node.js v20.13.1 documentation. *Node.js – Run JavaScript Everywhere*. URL: https://nodejs.org/docs/latest-v20.x/api/ (дата звернення: 24.05.2024).

3. Documentation | NestJS - A progressive Node.js framework. Documentation | NestJS - A progressive Node.js framework. URL: https://docs.nestjs.com/ (дата звернення: 24.05.2024).

4. Lombardi A. WebSocket: lightweight client-server communications. O'Reilly Media, Incorporated, 2015. 144 с.

5. Getting started | axios docs. *Axios*. URL: https://axios-http.com/docs/intro (дата звернення: 24.05.2024).

6. Miles R., Hamilton K. Learning UML 2. 0. O'Reilly Media, Incorporated, 2006. 303 с.

7. Nest.js – architectural pattern, controllers, providers, and modules. *Medium*. URL: https://medium.com/geekculture/nest-js-architectural-pattern-controllers-providers-and-modules-406d9b192a3a (дата звернення: 24.05.2024).

8. Трофіменко О. О. Використання ChatGPT для створення адаптивних діалогів у відеоіграх / О. О. Трофіменко // Радіоелектроніка та молодь у XXI столітті: тези доповідей 27-го Міжнародного молодіжного форуму, 10–12 травня 2023 р. – Харків : ХНУРЕ, 2023. – Т. 6. – С. 353–354.

9. Калініченко О. Ю. Імітація реалістичного ігрового супротивника за допомогою штучного інтелекту на основі моделі GPT Text-Davinci-003 / О. Ю. Калініченко // Радіоелектроніка та молодь у XXI столітті: тези доповідей 28-го Міжнародного молодіжного форуму, 16–18 квітня 2024 р. – Харків : ХНУРЕ, 2024. – Т. 6. – С. 363–365.

10. Mayo, Matthew. *Mastering Generative AI and Prompt Engineering: A Practical Guide for Data Scientists* / Matthew Mayo, KDnuggets Editor-in-Chief. – Data Science Horizons, 2023. – 41.

11. API Reference - OpenAI API. *OpenAI*. URL: https://platform.openai.com/docs/api-reference/introduction (дата звернення: 24.05.2024).

12. Hosting apps on render.com. *Medium*. URL: https://medium.com/@kanezi.com/hosting-apps-on-render-com-15edbdc18490 (дата звернення: 24.05.2024).

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

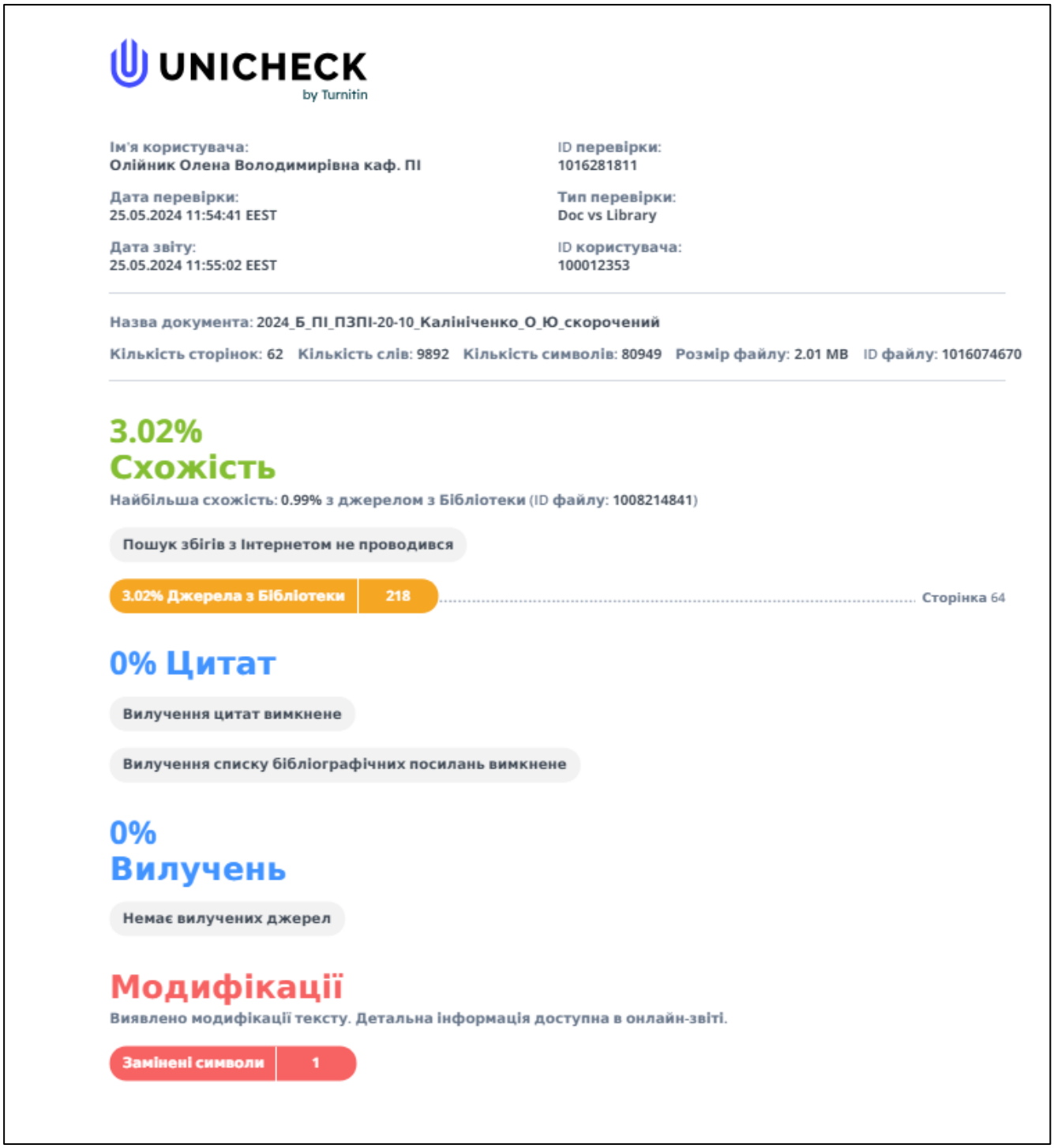


Рисунок А.1 – Результат перевірки на унікальність тексту в базі ХНУРЕ (знімок екрана виконано самостійно)

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)

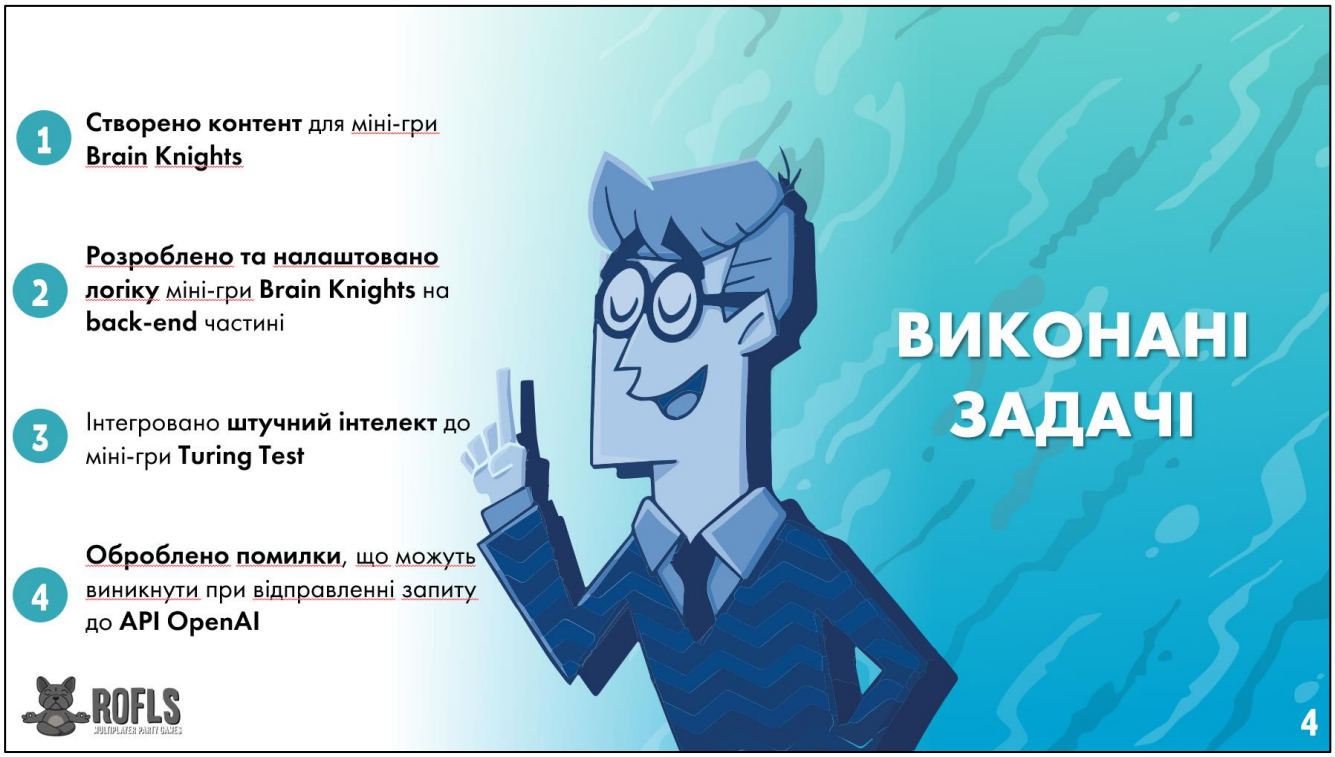


Рисунок Б.4 – Четвертий слайд презентації (знімок екрана виконано самостійно)

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

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

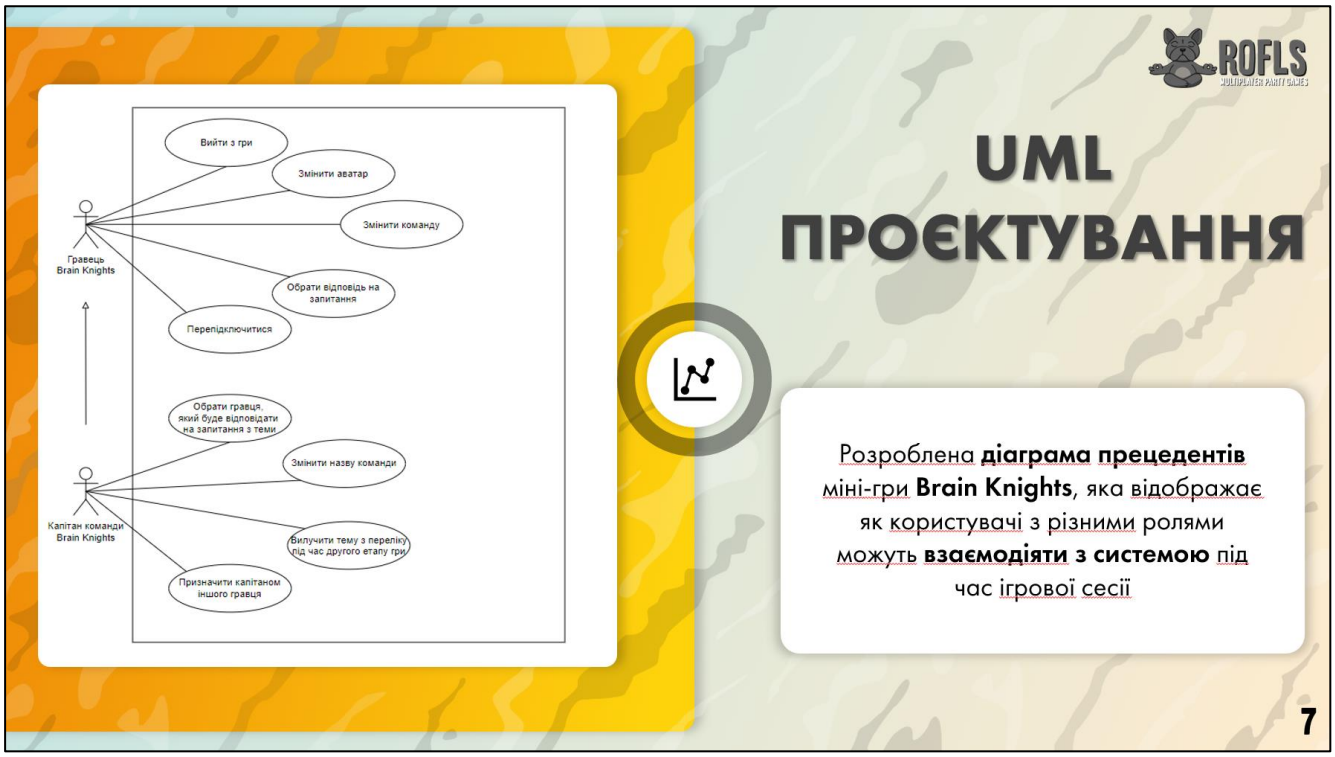


Рисунок Б.7 – Сьомий слайд презентації (знімок екрана виконано самостійно)

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

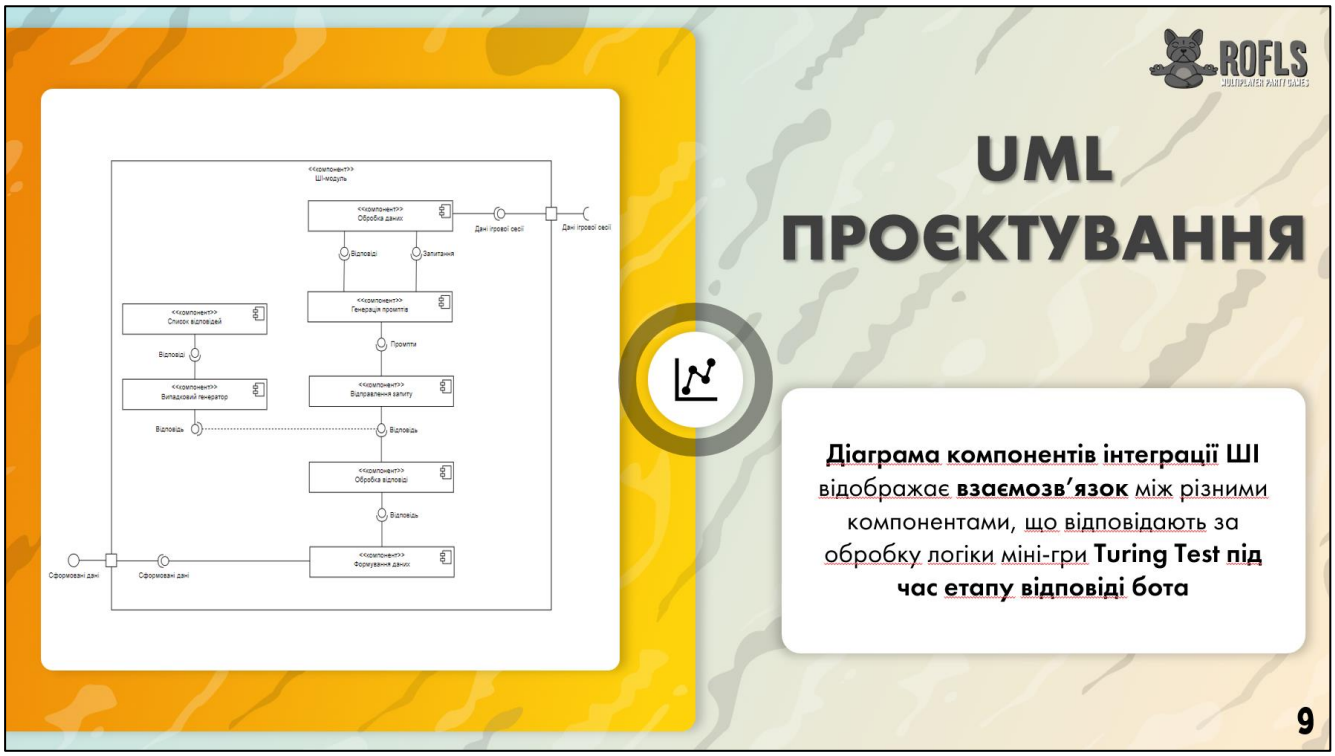


Рисунок Б.9 – Дев’ятий слайд презентації (знімок екрана виконано самостійно)

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

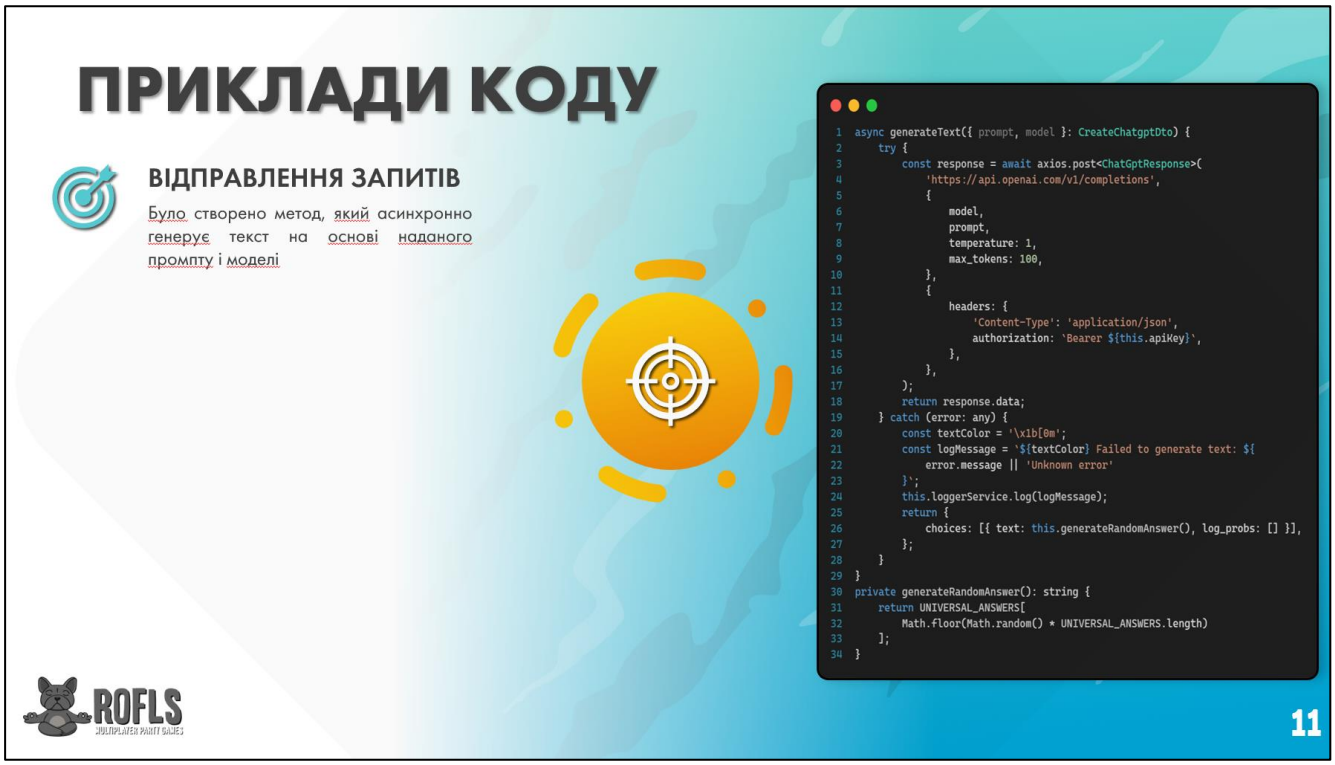


Рисунок Б.11 – Одинадцятий слайд презентації (знімок екрана виконано самостійно)

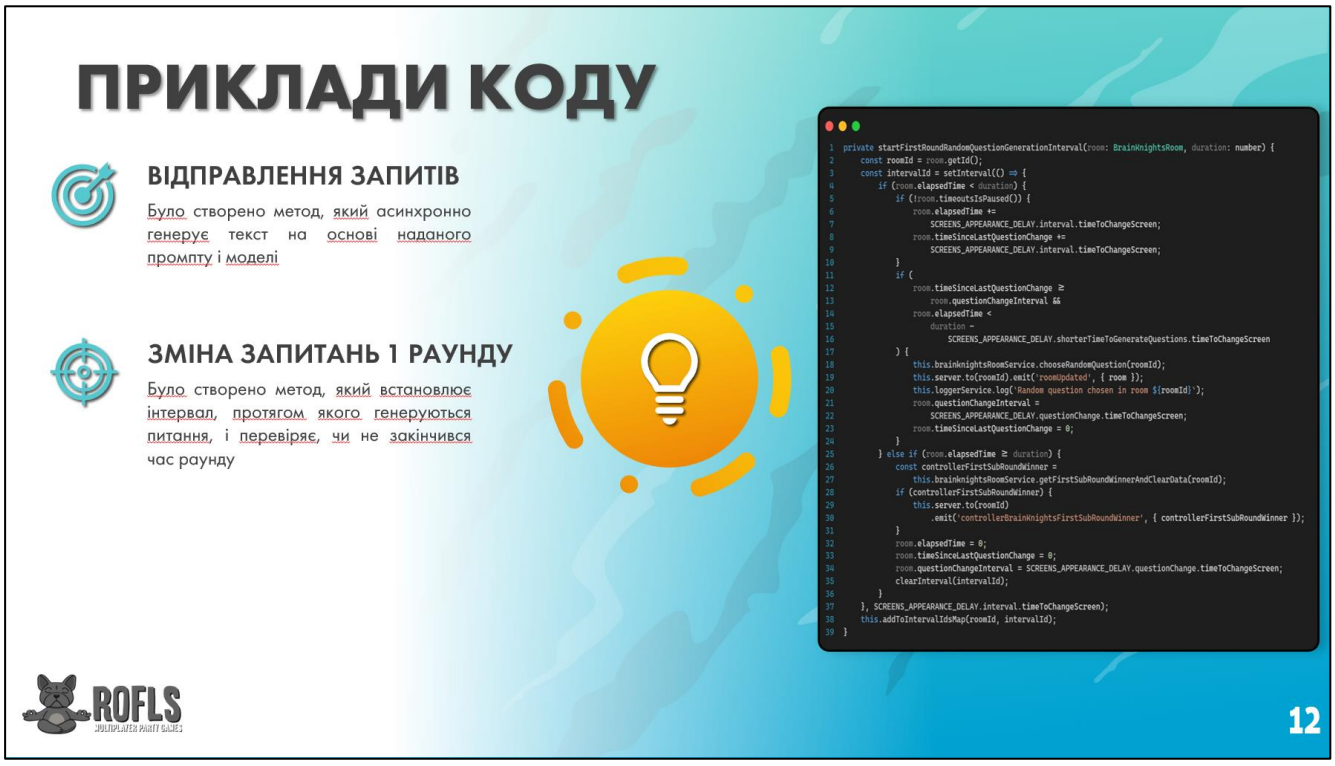


Рисунок Б.12 – Дванадцятий слайд презентації (знімок екрана виконано самостійно)

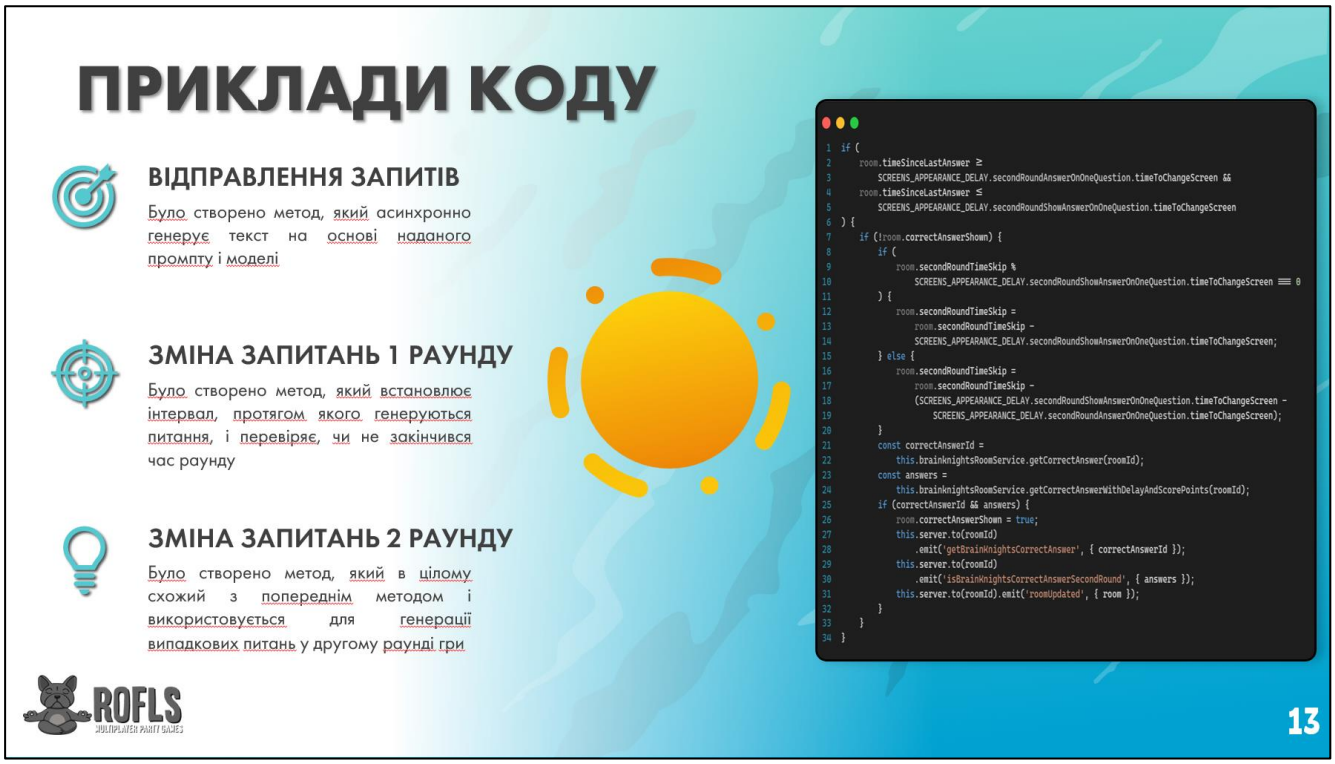


Рисунок Б.13 – Тринадцятий слайд презентації (знімок екрана виконано самостійно)

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

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

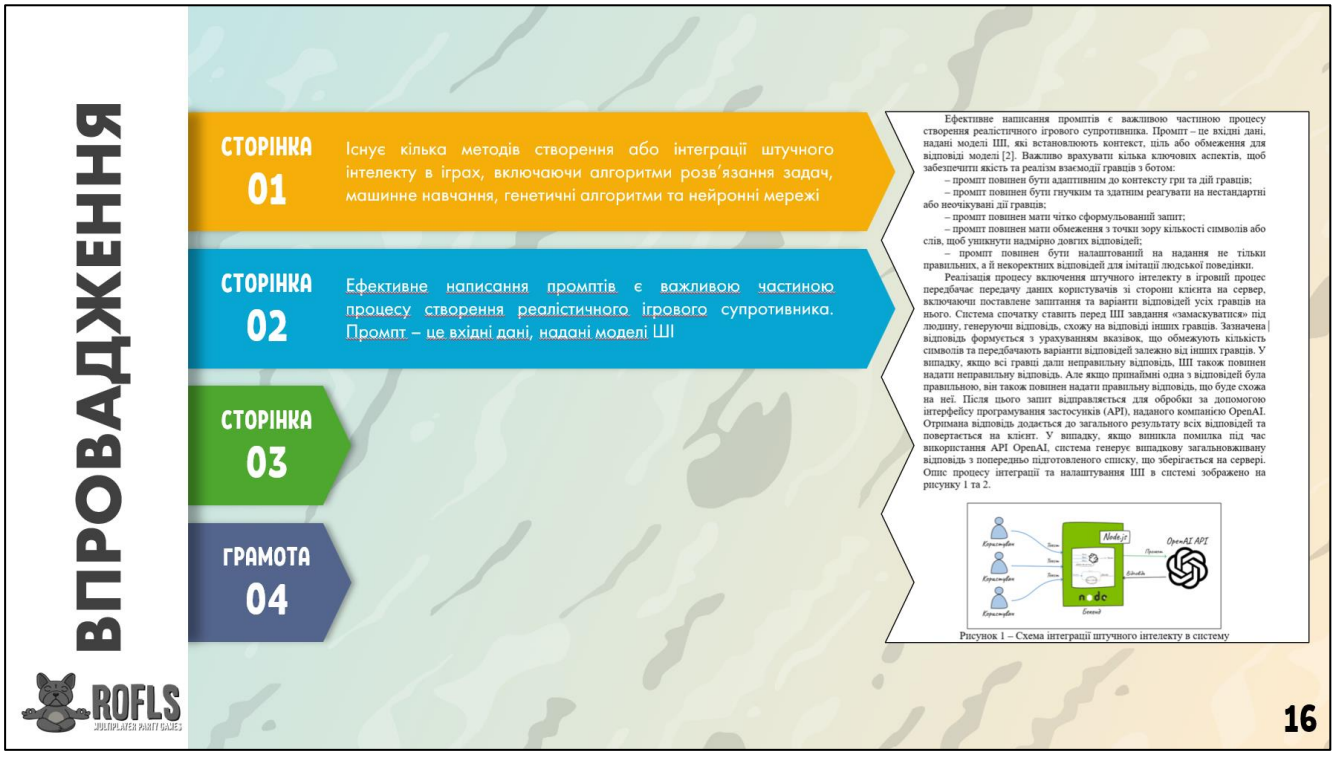


Рисунок Б.16 – Шістнадцятий слайд презентації (знімок екрана виконано самостійно)

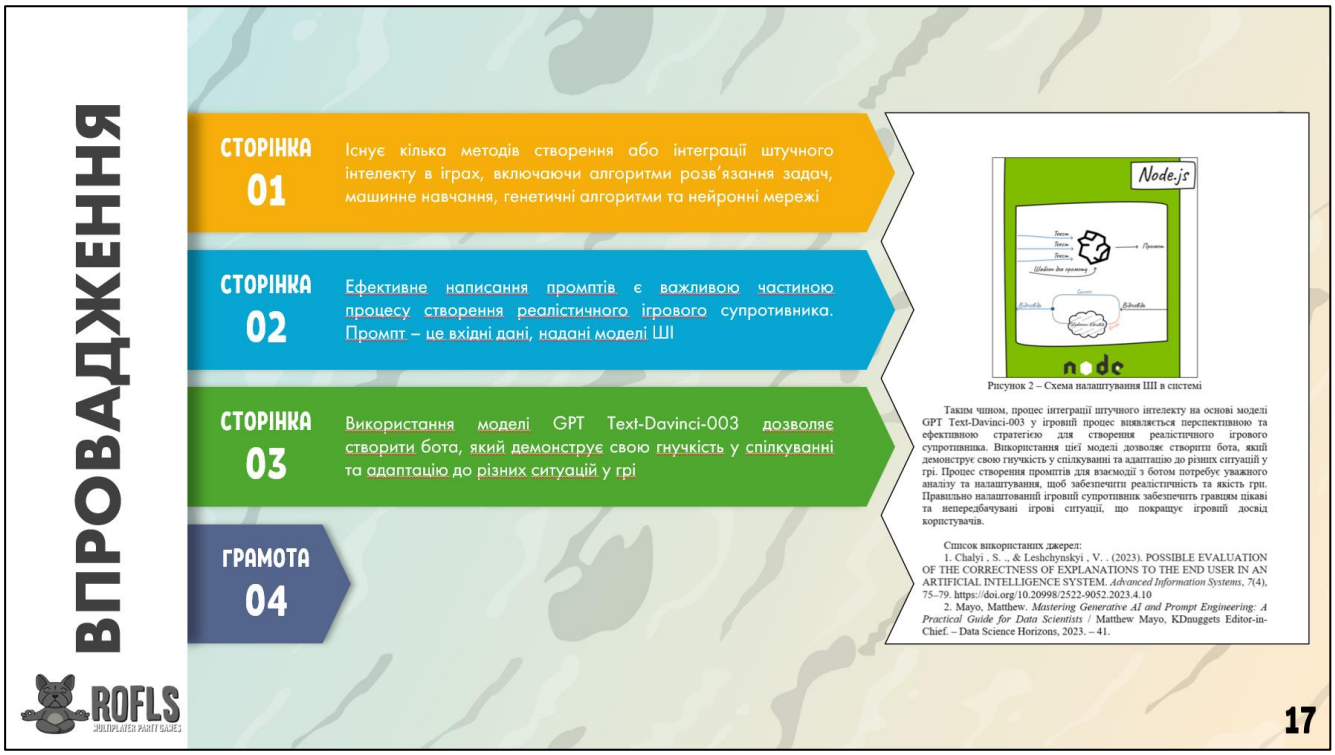


Рисунок Б.17 – Сімнадцятий слайд презентації (знімок екрана виконано самостійно)

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

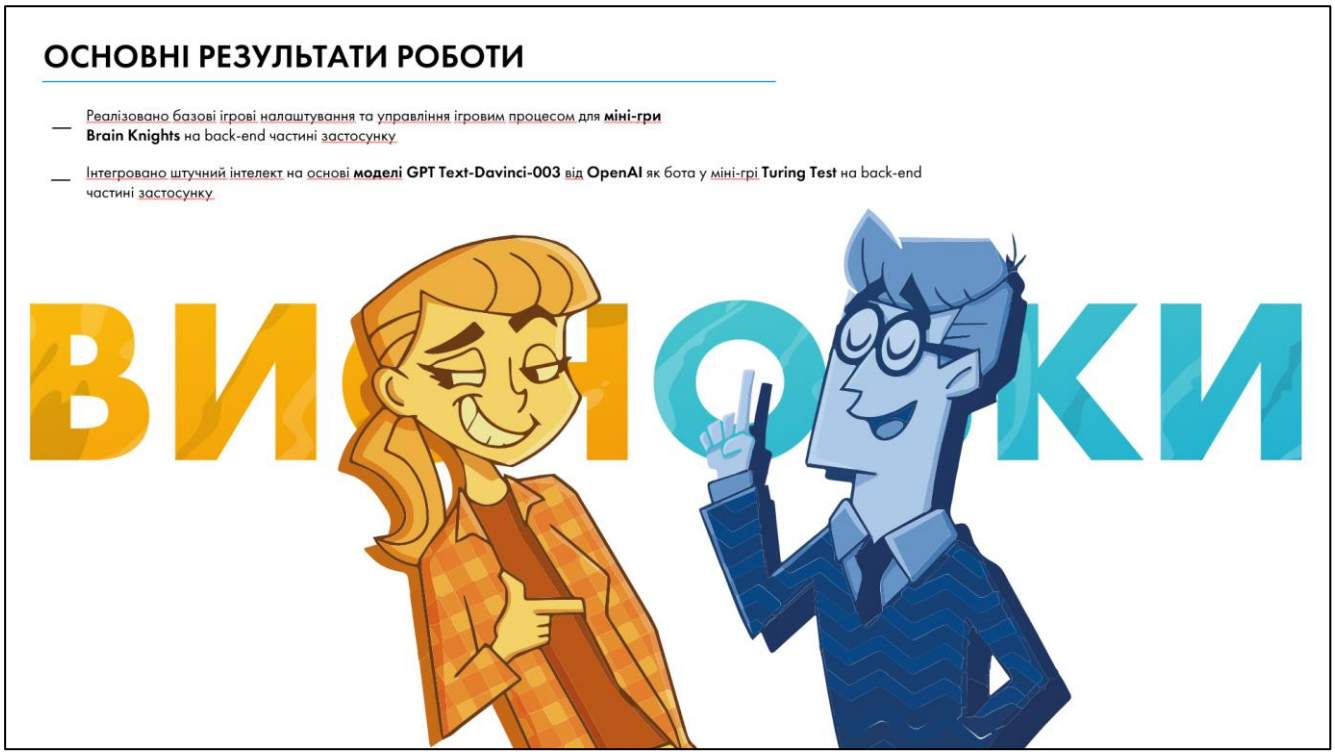


Рисунок Б.19 – Дев’ятнадцятий слайд презентації (знімок екрана виконано самостійно)

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.

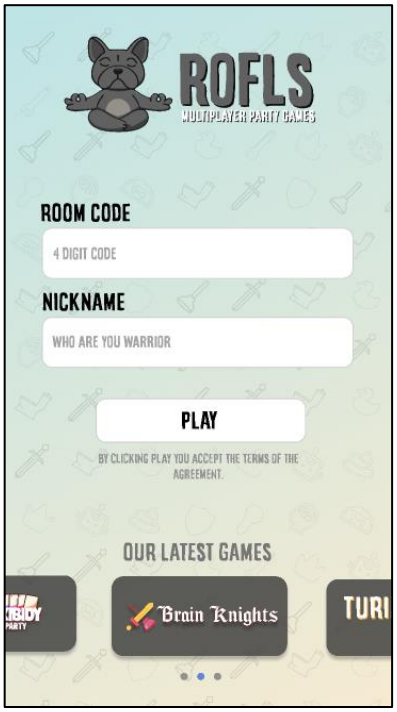


Рисунок 3.2 – Інтерфейс стартового екрану контролера (знімок екрана виконано самостійно)

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.

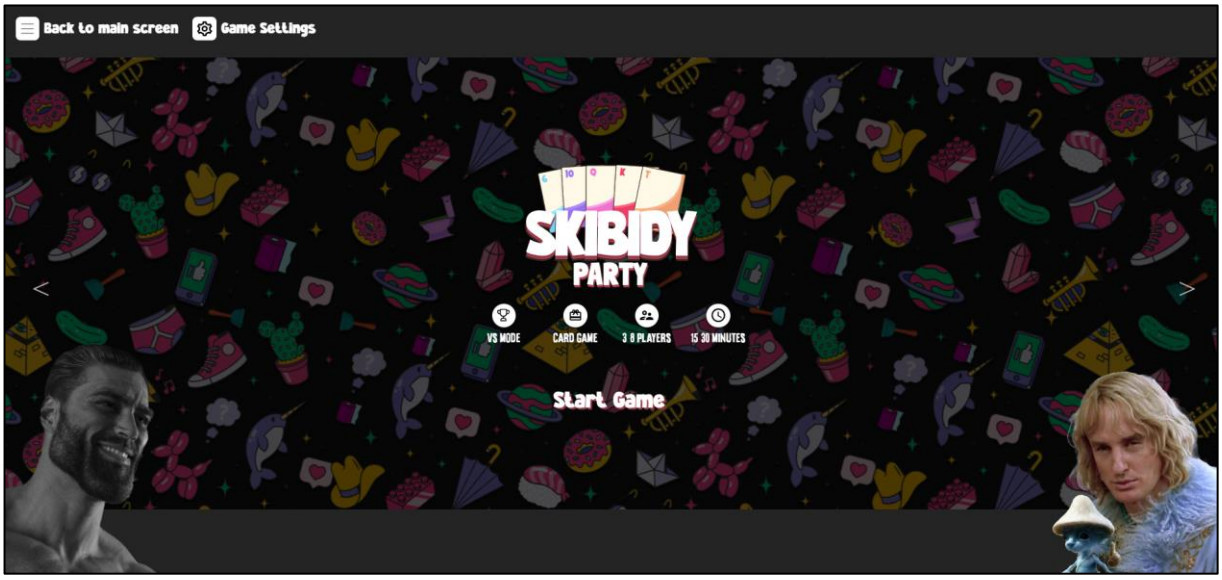


Рисунок 3.4 – Вибір гри Skibidy Party (знімок екрана виконано самостійно)

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

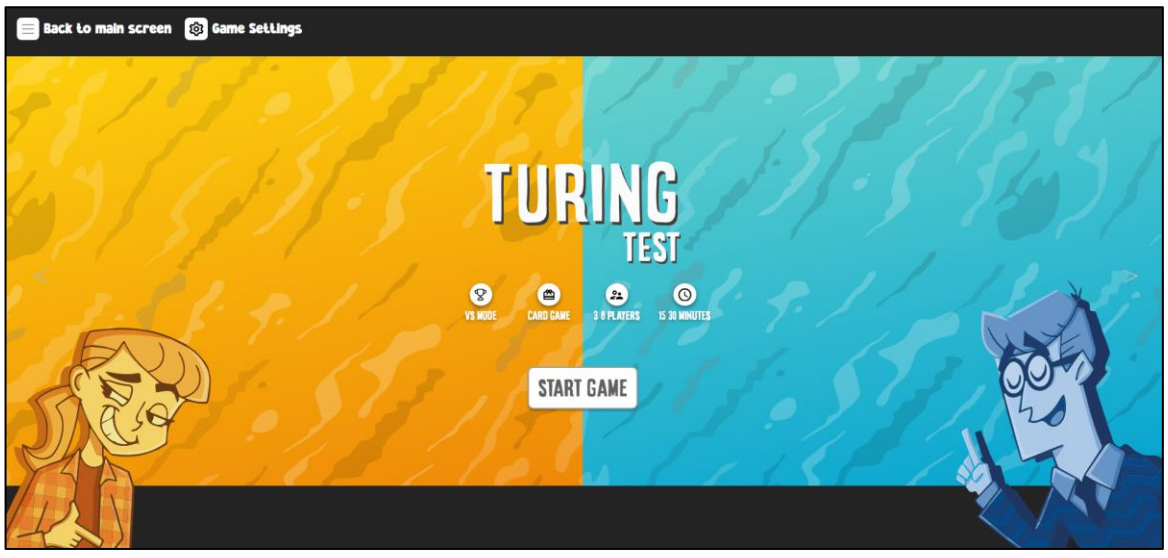


Рисунок 3.5 – Вибір гри Turing Test (знімок екрана виконано самостійно)

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

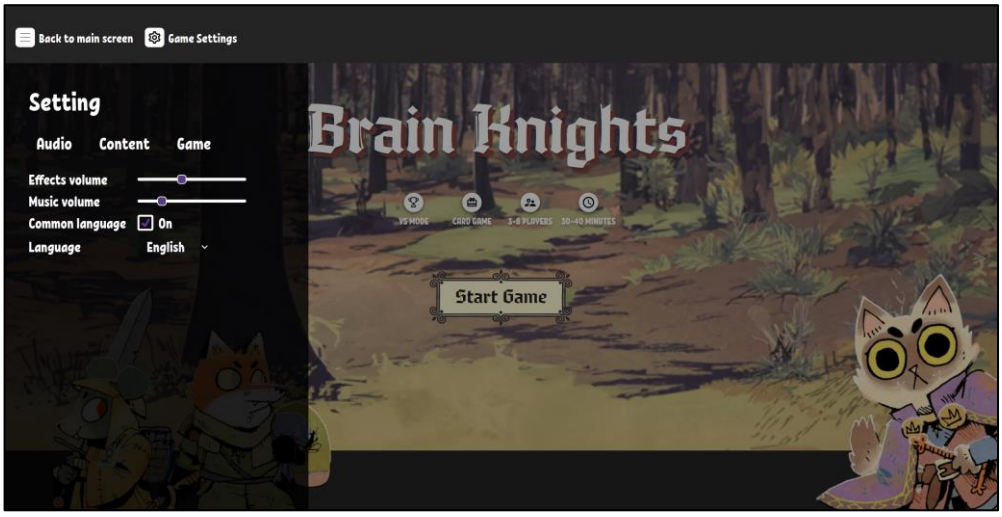


Рисунок 3.6 – Налаштуванні ігрового процесу (знімок екрана виконано самостійно)

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.

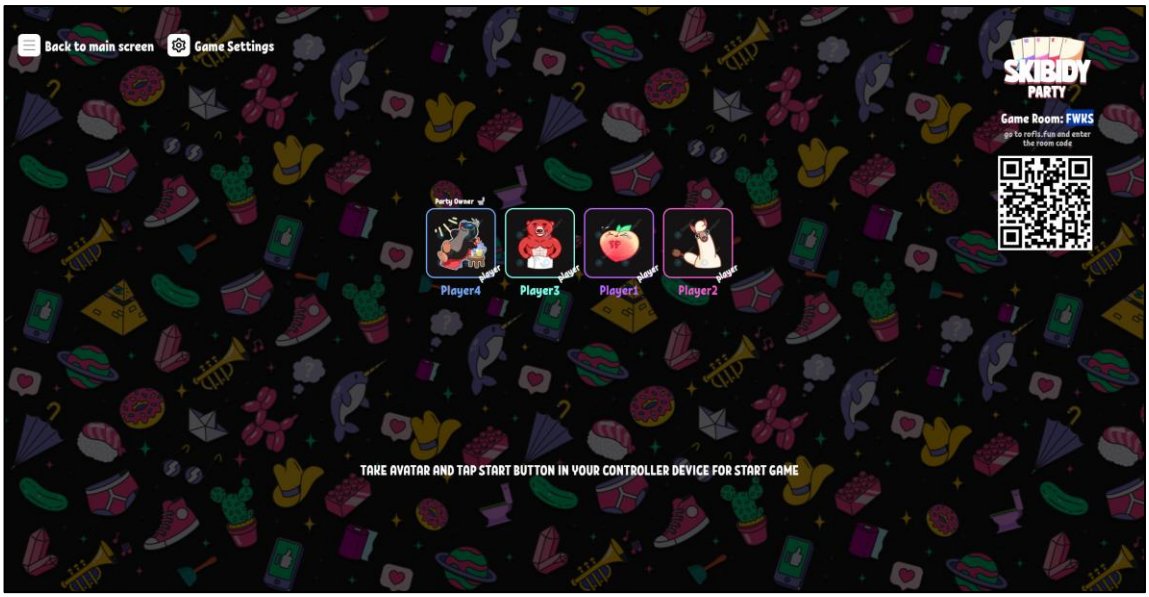


Рисунок 3.7 – Інтерфейс ігрової кімнати в грі Skibidy Party на хості (знімок екрана виконано самостійно)

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

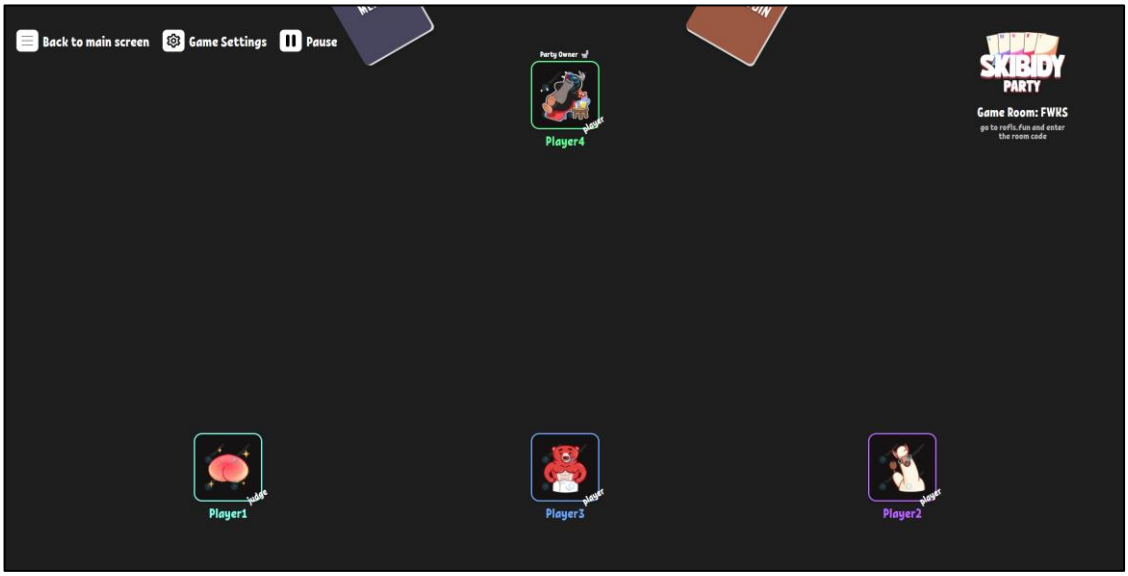


Рисунок 3.8 – Інтерфейс ігрового поля в Skibidy Party на хості (знімок екрана виконано самостійно)

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

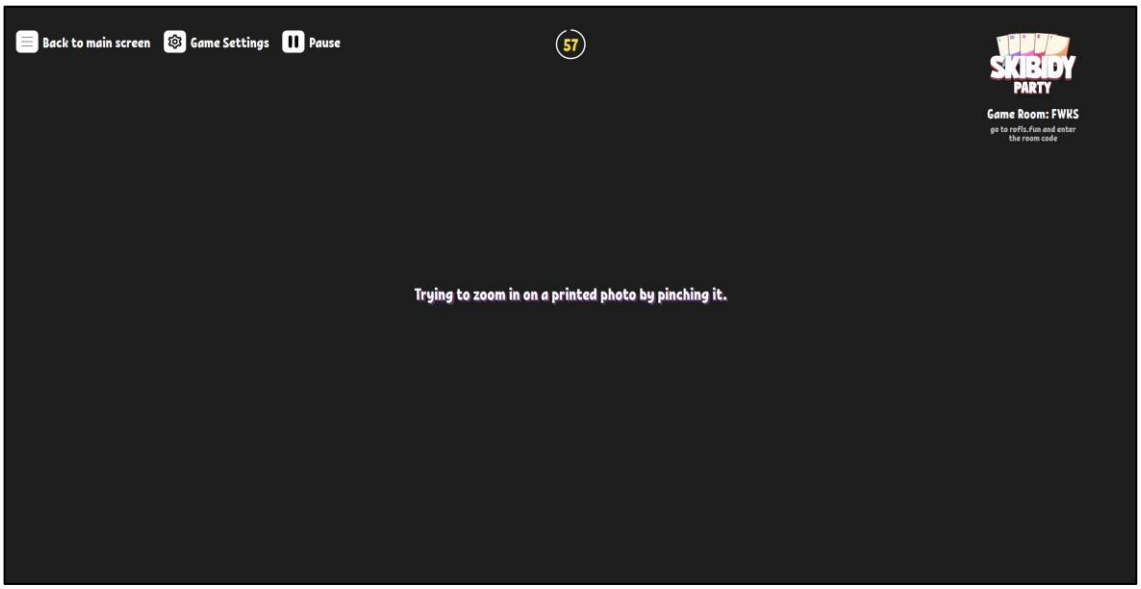


Рисунок 3.9 – Інтерфейс ігрової ситуації в Skibidy Party на хості (знімок екрана виконано самостійно)

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

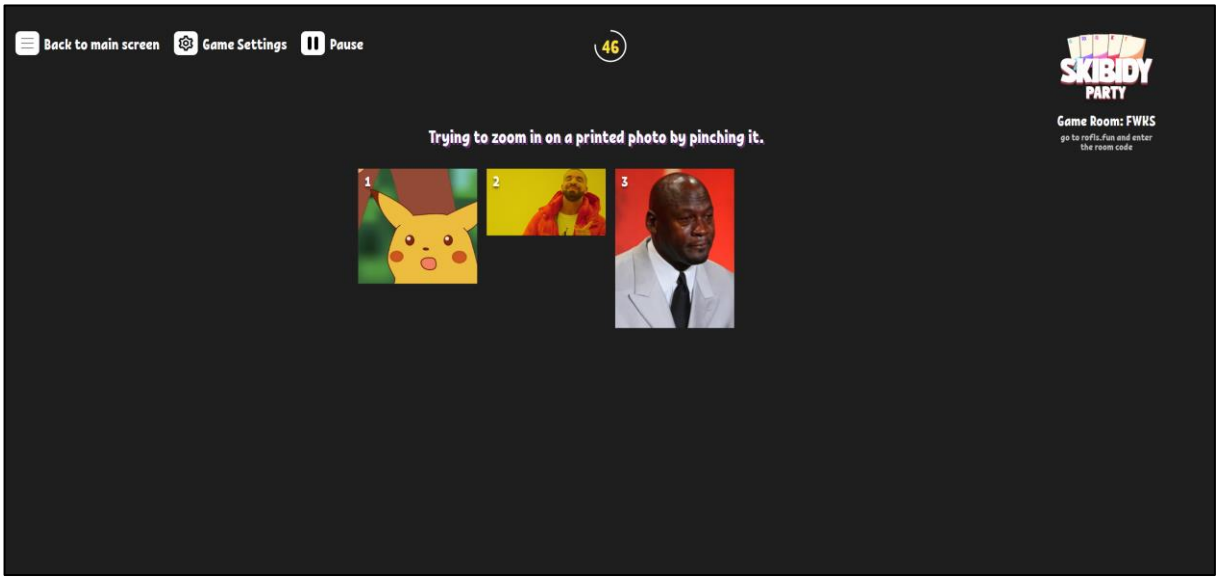


Рисунок 3.10 – Інтерфейс обраних мемів в Skibidy Party на хості (знімок екрана виконано самостійно)

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

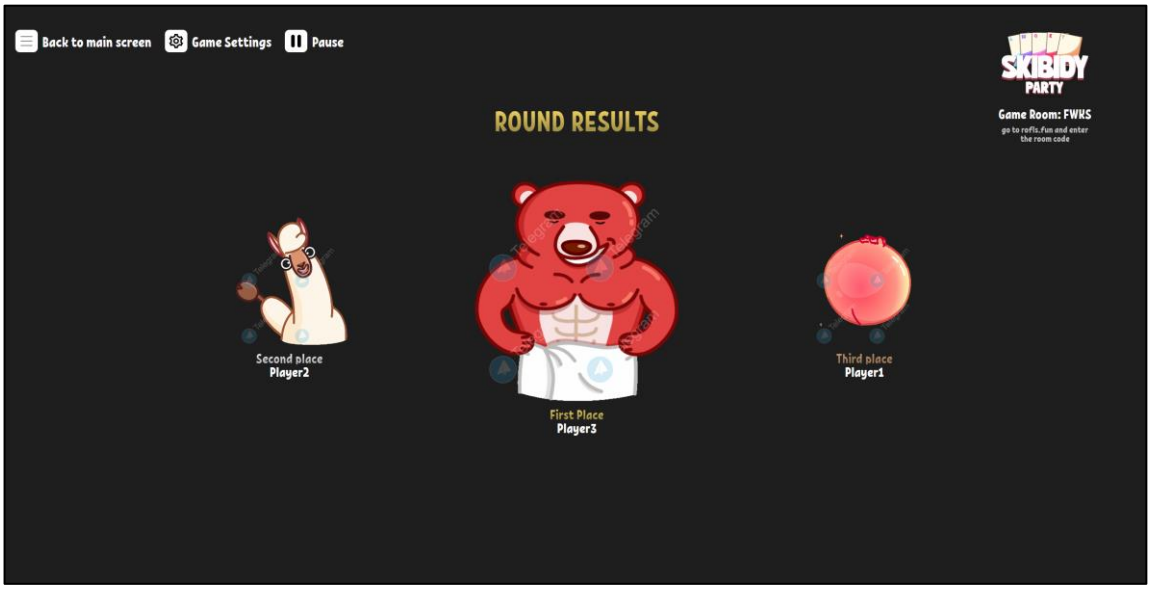


Рисунок 3.11 – Інтерфейс переможців раунду в Skibidy Party на хості (знімок екрана виконано самостійно)

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.

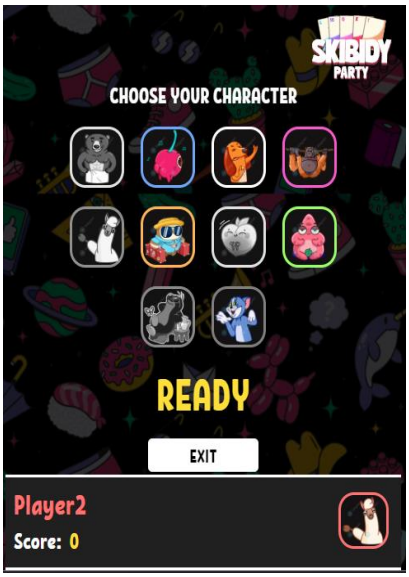


Рисунок 3.13 – Вибір аватару в Skibidy Party на контролері (знімок екрана виконано самостійно)

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

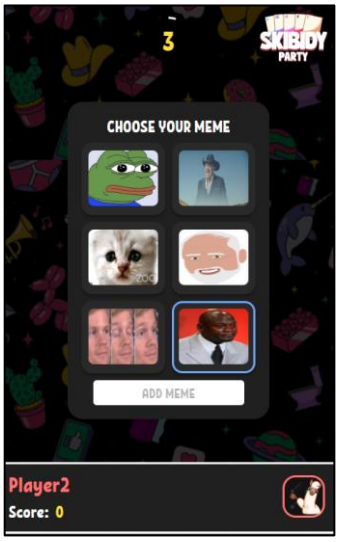


Рисунок 3.14 – Вибір мему в Skibidy Party на контролері (знімок екрана виконано самостійно)

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

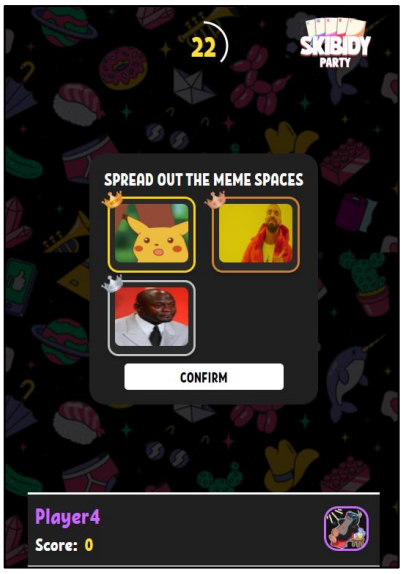


Рисунок 3.15 – Вибір переможного мему в Skibidy Party на контролері (знімок екрана виконано самостійно)

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.

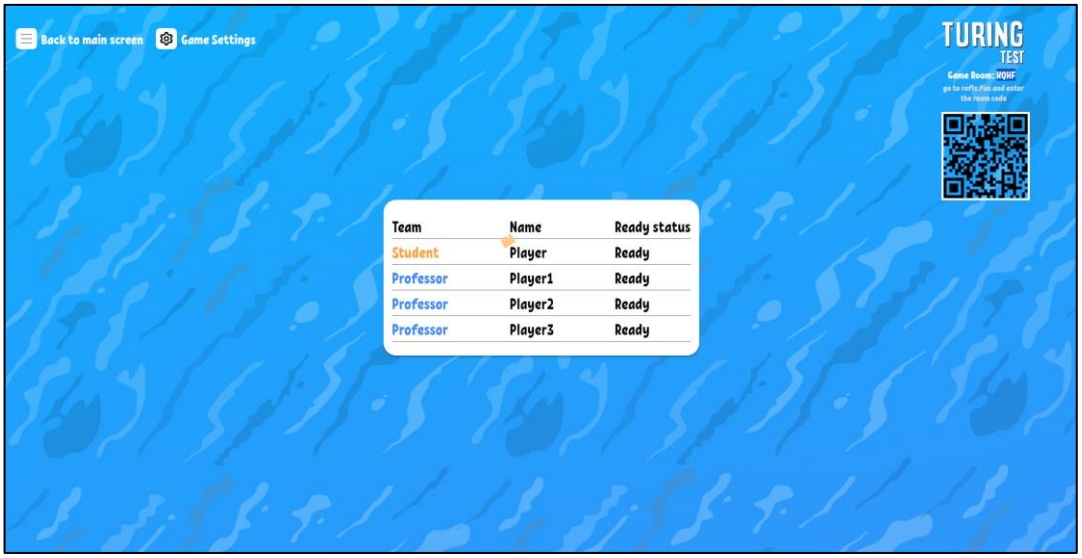


Рисунок 3.16 – Інтерфейс ігрової кімнати в грі Turing Test на хості (знімок екрана виконано самостійно)

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.

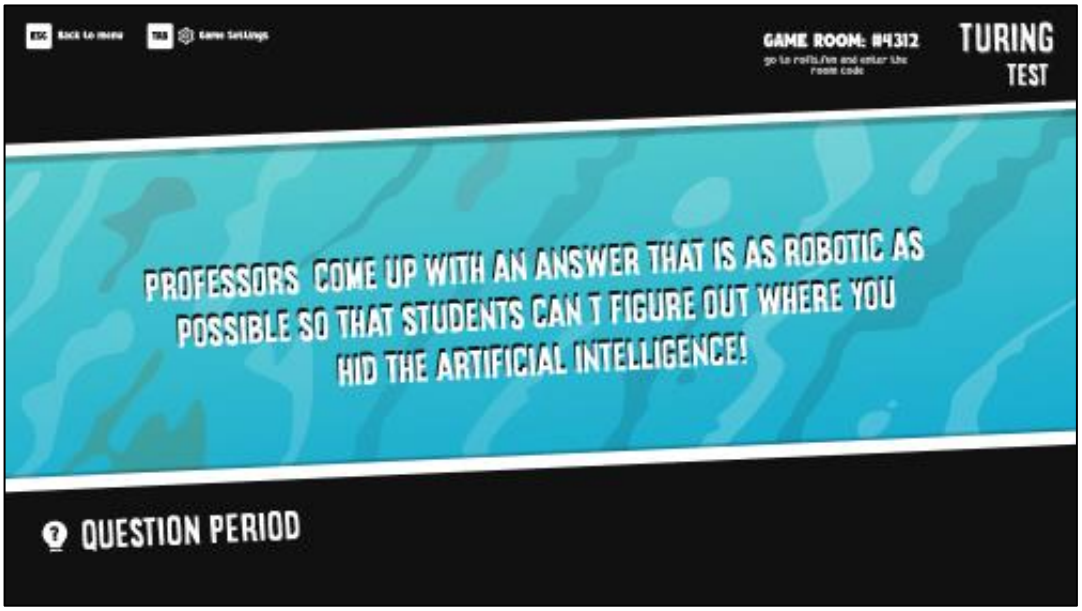


Рисунок 3.18 – Інтерфейс раунду в грі Turing Test на хості (знімок екрана виконано самостійно)

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.

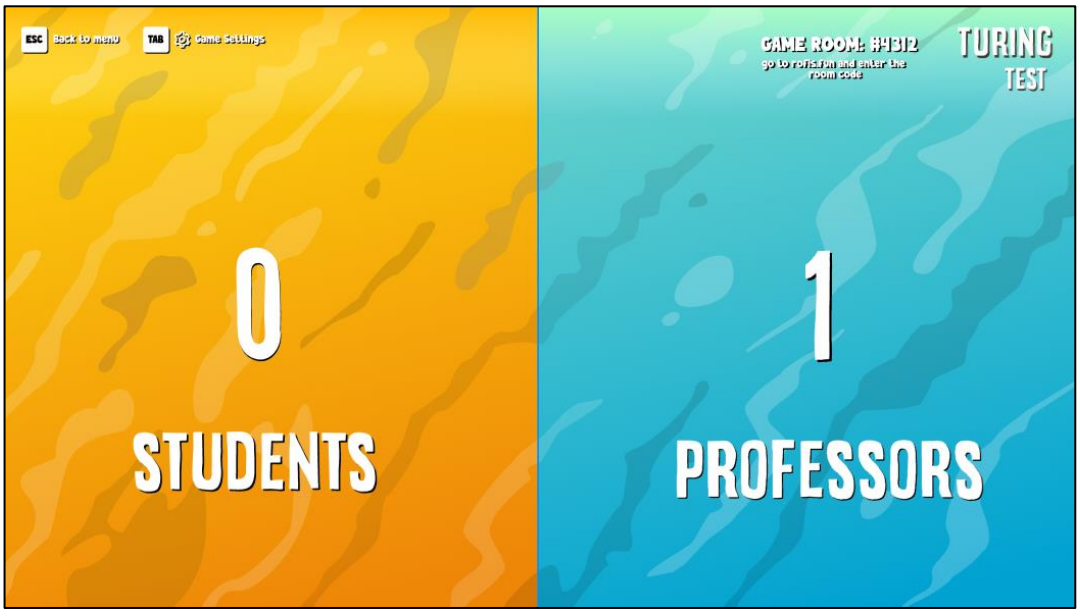


Рисунок 3.23 – Інтерфейс рахунку в грі Turing Test на хості (знімок екрана виконано самостійно)

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.

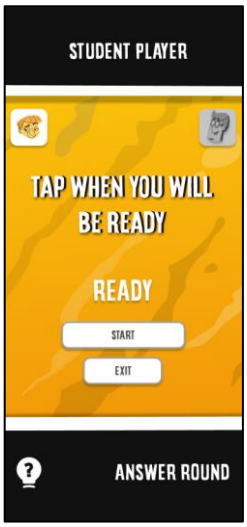


Рисунок 3.24 – Інтерфейс вибору команди в грі Turing Test на котролері (знімок екрана виконано самостійно)

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».

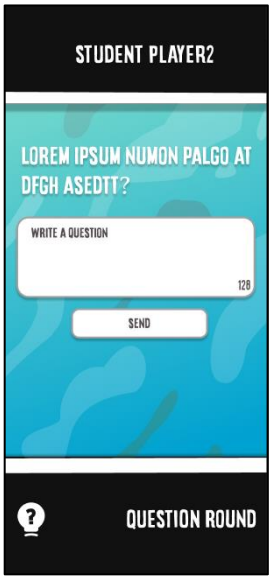


Рисунок 3.26 – Інтерфейс написання відповіді на питання професорами в грі Turing Test на котролері (знімок екрана виконано самостійно)

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.

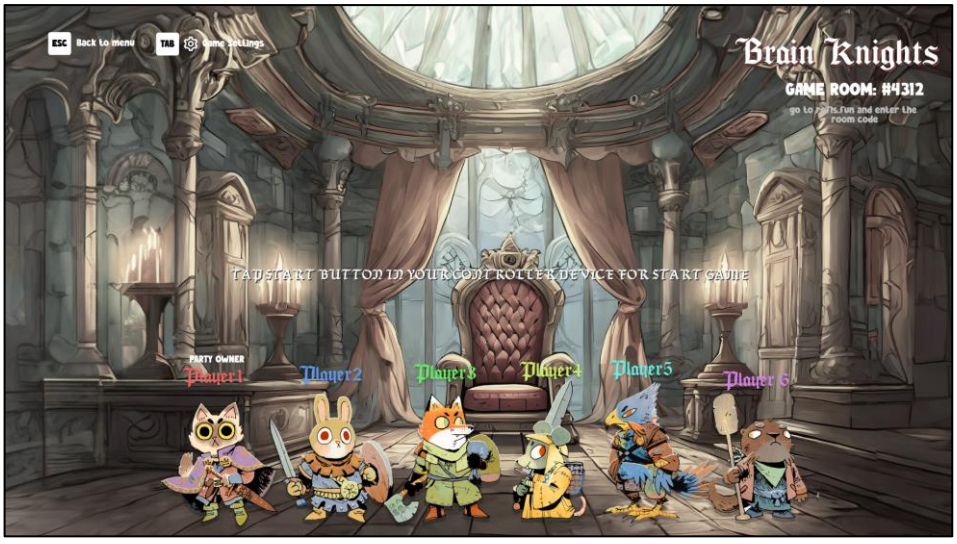


Рисунок 3.29 – Інтерфейс ігрової кімнати на головному екрані гри в грі Brain Knights на хості (знімок екрана виконано самостійно)

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

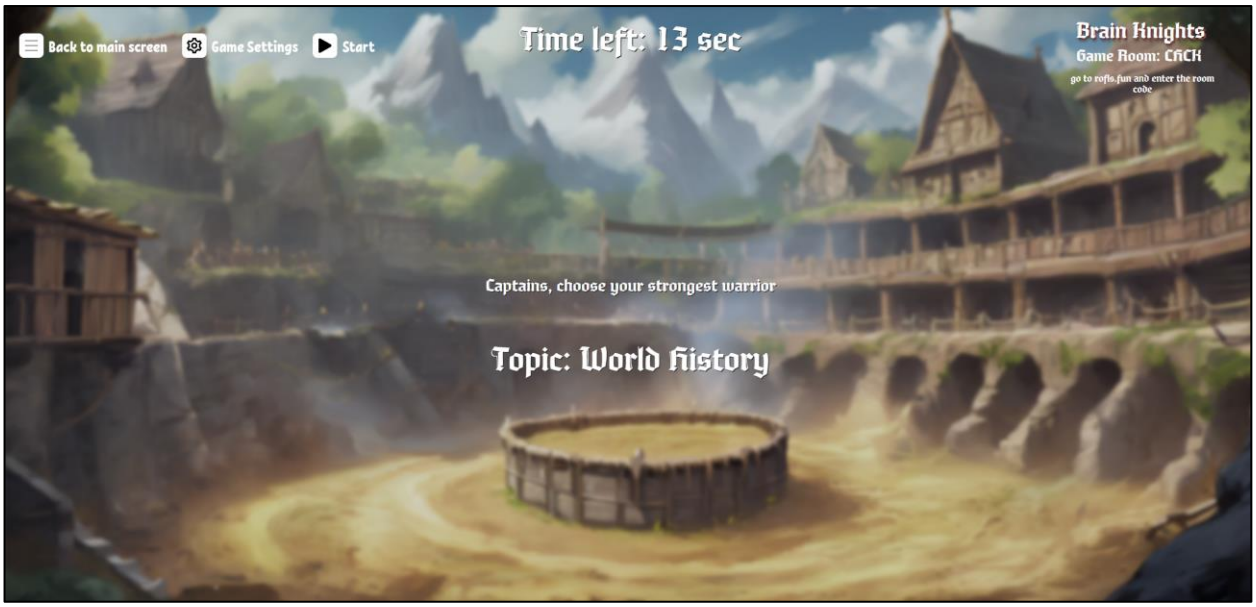


Рисунок 3.30 – Інтерфейс теми раунду в грі Brain Knights на хості (знімок екрана виконано самостійно)

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

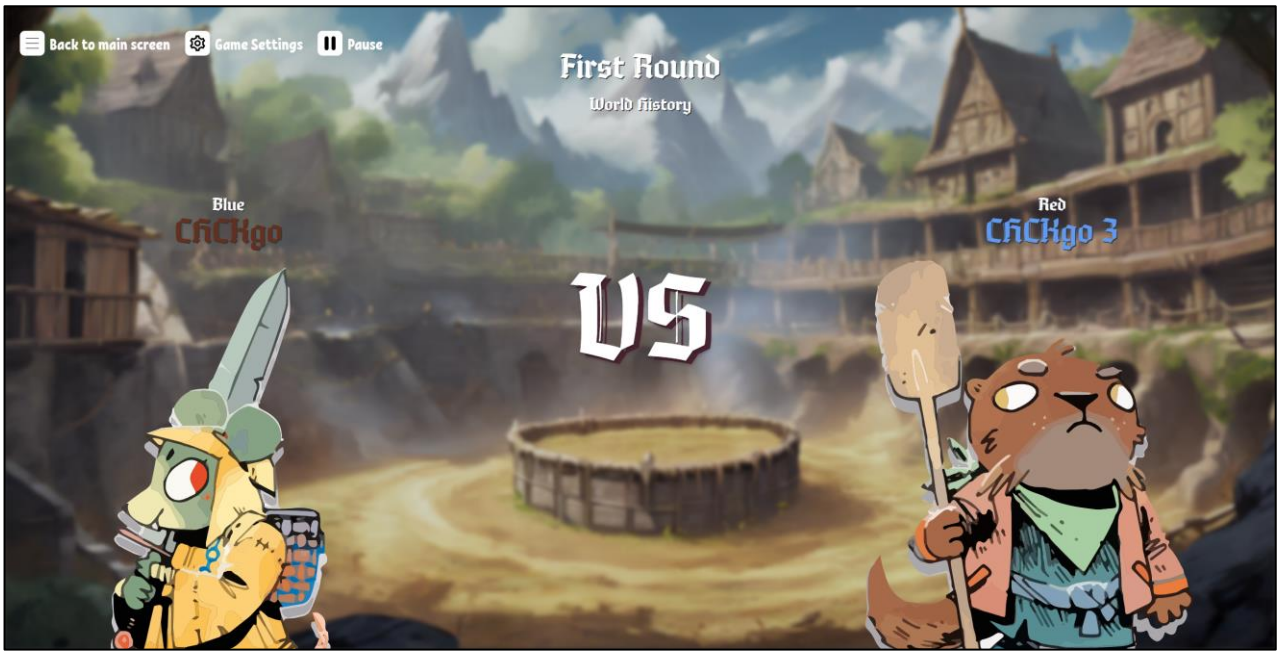


Рисунок 3.31 – Інтерфейс початку бліц-раунду в грі Brain Knights на хості (знімок екрана виконано самостійно)

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

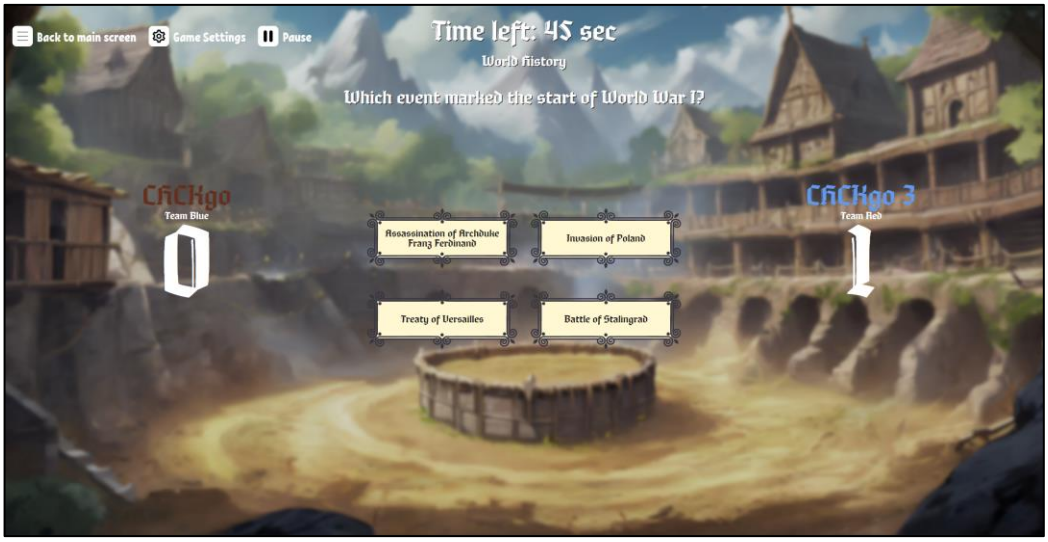


Рисунок 3.32 – Інтерфейс ігрового раунду в грі Brain Knights на хості (знімок екрана виконано самостійно)

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.

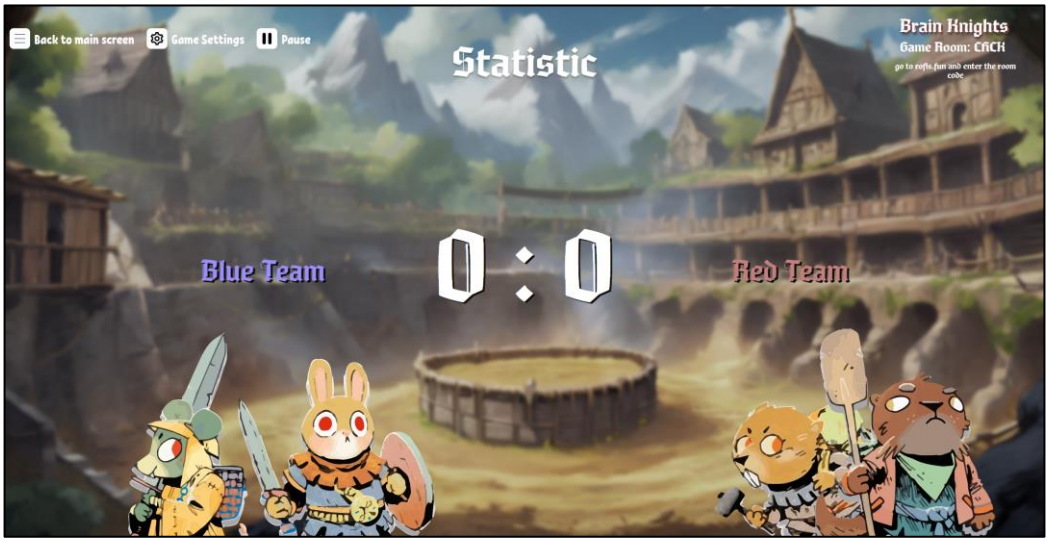


Рисунок 3.34 – Проміжні результати раунду в грі Brain Knights на хості (знімок екрана виконано самостійно)

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.

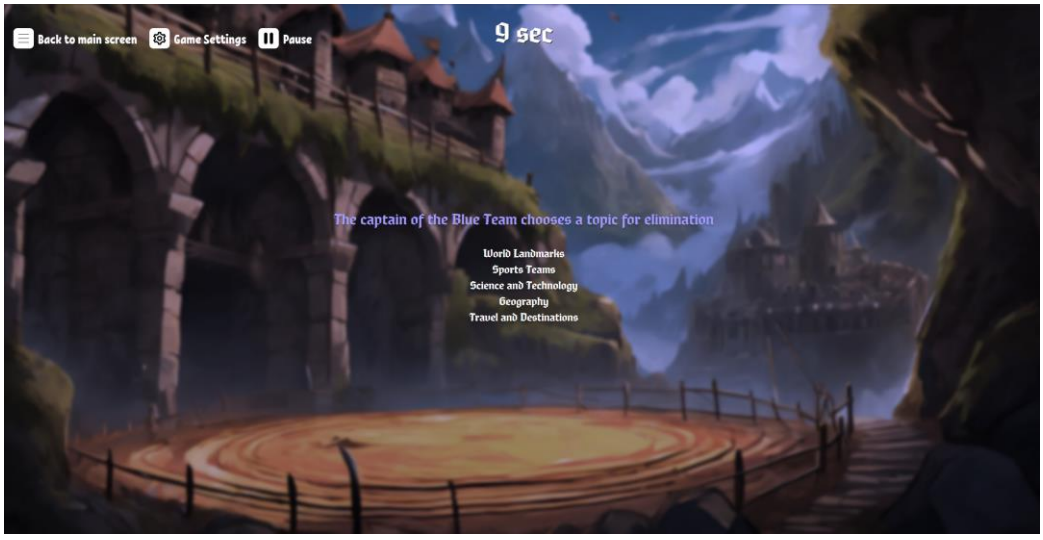


Рисунок 3.35 – Вибір теми для другої стадії в грі Brain Knights на хості (знімок екрана виконано самостійно)

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

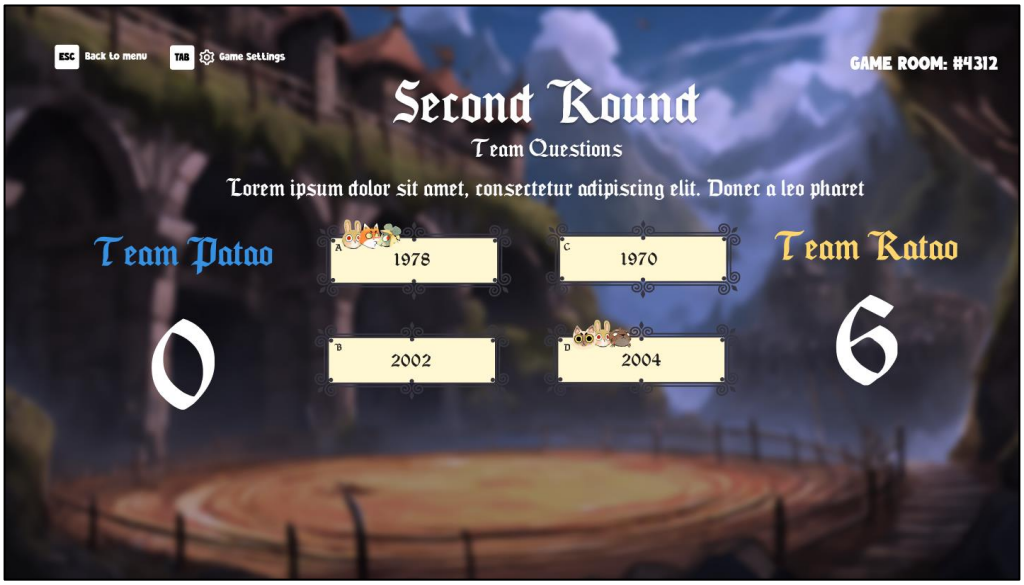


Рисунок 3.36 – Ігровий раунд у другій стадії в грі Brain Knights на хості (знімок екрана виконано самостійно)

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.

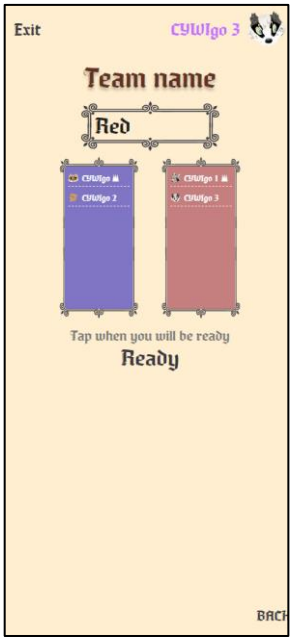


Рисунок 3.38 – Вибір команди в грі Brain Knights на хості (знімок екрана виконано самостійно)

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

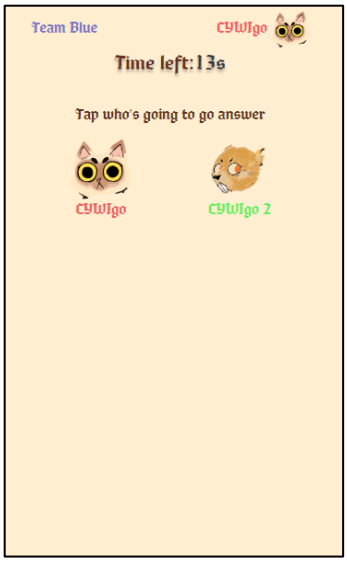


Рисунок 3.39 – Вибір гравця для відповіді в грі Brain Knights на хості (знімок екрана виконано самостійно)

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

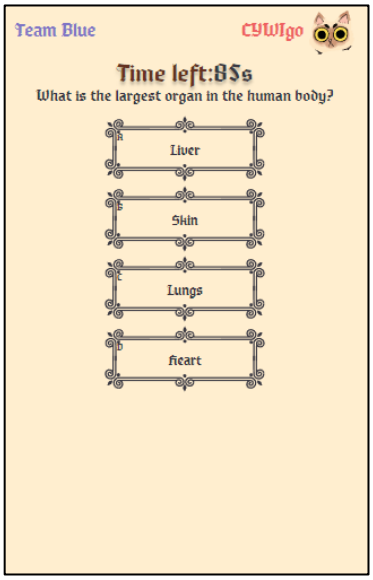


Рисунок 3.40 – Вибір відповіді на запитання в грі Brain Knights на хості (знімок екрана виконано самостійно)

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 немає.

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:	Формування відповіді на запитання студента на основі готових відповідей професорів за допомогою штучного інтелекту у грі Turing Test		
Власник тесту:	Калініченко Олександр Юрійович		
Дата створення:	20.05.2024		
Мета тесту:	Перевірити коректність формування відповіді штучним інтелектом на основі отриманого запитання студента та отриманих відповідей професорів		
Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	З використанням клієнтської частини створено ігрову сесію гри Turing Test	Ігрову сесію створено, і її дані збережено у пам’яті серверного застосунку	Пройдено
2	З використанням клієнтської частини до створеної кімнати підключено чотирьох гравців	Серверний застосунок записав у лог подію підключення кожного гравця, а дані гравців додаються до масиву та зберігаються у пам’яті застосунку	Пройдено

Кінець таблиці Д.1

Передумова			
№	Опис випадку	Очікуваний результат	Висновок
3	З використанням клієнтської частини підтверджено готовність усіх гравців до гри	Серверний застосунок записав у лог подію зміни готовності до гри кожного гравця, а дані готовності гравців були змінені та збережені у пам’яті застосунку	Пройдено
4	Власник кімнати розпочав гру, використовуючи клієнтську частину	Серверний застосунок записав у лог подію початку гри та змінив статус гри у пам’яті застосунку	Пройдено
Команда професорів надіслала свої відповіді у грі Turing Test			
1	З використанням клієнтської частини гравці команди студентів надіслали свої запитання, а гравці команди професорів – свої відповіді	Серверний застосунок зберіг отримані дані відповідей та запитань у пам’яті, сформував промпти для відправки до API OpenAI, надіслав їх до моделі GPT Text-Davinci-003, обробив отримані відповіді та зберіг їх у пам’яті. У разі помилки від API, застосунок випадково обрав заготовлену відповідь зі списку та зберіг її у пам’яті застосунку	Пройдено
Результати тестування			
Тестувальник: Калініченко О. Ю.	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:	Зміна статусу гравця після виходу з розпочатою гри		
Власник тесту:	Калініченко Олександр Юрійович		
Дата створення:	20.05.2024		
Мета тесту:	Перевірити коректність обробки запиту на зміну статусу гравця після виходу		
Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	З використанням клієнтської частини створено ігрову сесію гри	Ігрову сесію створено, і її дані збережено у пам’яті серверного застосунку	Пройдено
2	З використанням клієнтської частини до створеної кімнати підключено чотирьох гравців	Серверний застосунок записав у лог подію підключення кожного гравця, а дані гравців додаються до масиву та зберігаються у пам’яті застосунку	Пройдено
3	З використанням клієнтської частини підтверджено готовність усіх гравців до гри	Серверний застосунок записав у лог подію зміни готовності до гри кожного гравця, а дані готовності гравців були змінені та збережені у пам’яті застосунку	Пройдено
4	Власник кімнати розпочав гру, використовуючи клієнтську частину	Серверний застосунок записав у лог подію початку гри та змінив статус гри у пам’яті застосунку	Пройдено

Кінець таблиці Д.2

Відключення гравця від ігрової сесії			
1	З використанням клієнтської частини гравець відключається від ігрової сесії	Серверний застосунок змінив статус підключення гравця з «Connected» на «Disconnected», записав у лог подію відключення та зберіг цю інформацію у пам’яті	Пройдено
Результати тестування			
Тестувальник: Калініченко О. Ю.	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:	Збереження інформації про гравців що відповідали в першому раунді гри Brain Knights
Власник тесту:	Калініченко Олександр Юрійович
Дата створення:	20.05.2024
Мета тесту:	Перевірити коректність роботи функції зберігання інформації про гравців що відповідали в першому раунді гри Brain Knights

Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	З використанням клієнтської частини створено ігрову сесію гри Brain Knights	Ігрову сесію створено, і її дані збережено у пам’яті серверного застосунку	Пройдено

Продовження таблиці Д.3

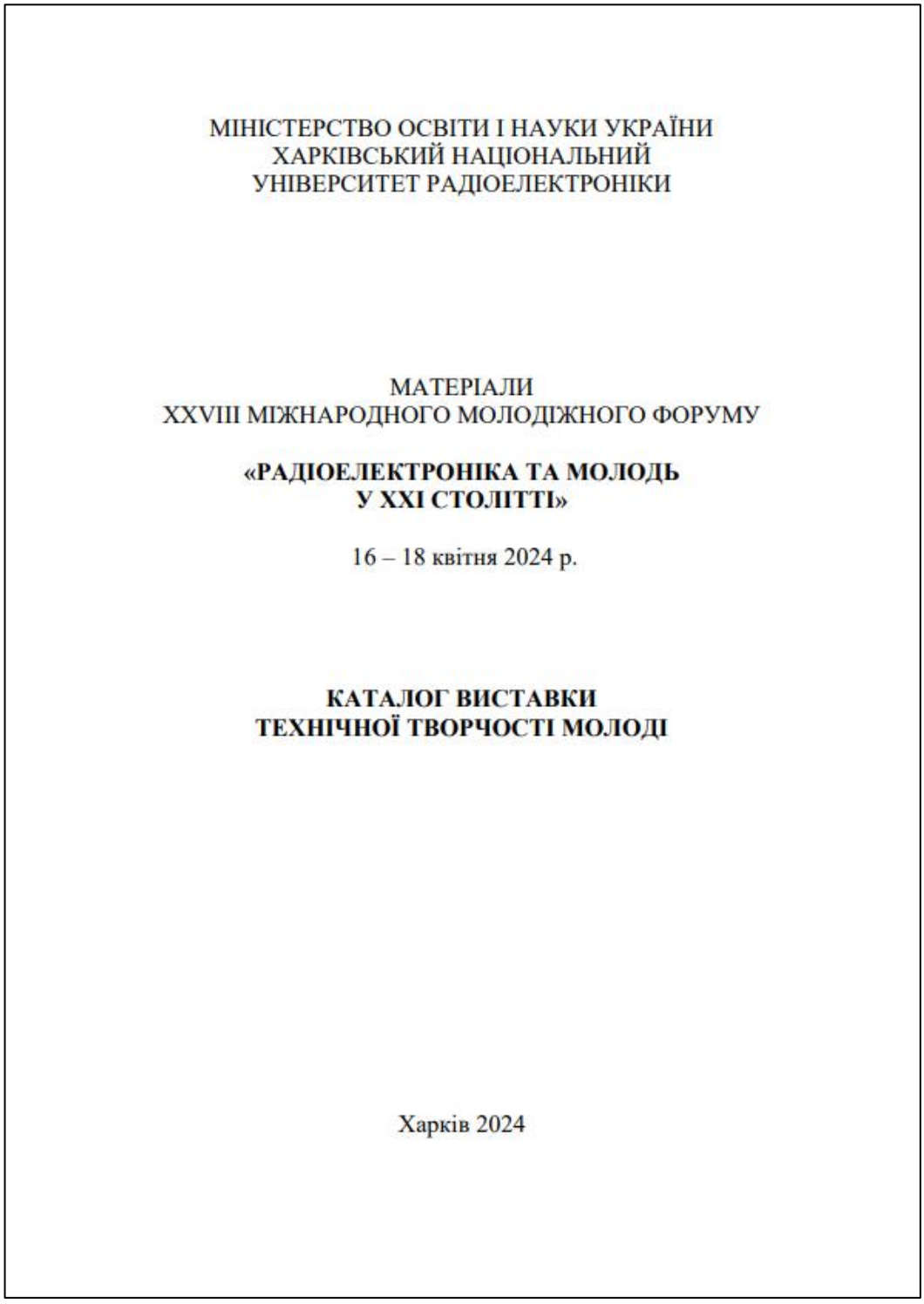
Передумова			
№	Опис випадку	Очікуваний результат	Висновок
2	З використанням клієнтської частини до створеної кімнати підключено чотирьох гравців	Серверний застосунок записав у лог подію підключення кожного гравця, а дані гравців додаються до масиву та зберігаються у пам’яті застосунку	Пройдено
3	З використанням клієнтської частини підтверджено готовність усіх гравців до гри	Серверний застосунок записав у лог подію зміни готовності до гри кожного гравця, а дані готовності гравців були змінені та збережені у пам’яті застосунку	Пройдено
4	Власник кімнати розпочав гру, використовуючи клієнтську частину	Серверний застосунок записав у лог подію початку гри та змінив статус гри у пам’яті застосунку	Пройдено
Вибір відповідача у бліц-опитуванні в першому раунді гри Brain Knights			
1	Капітан команди обирає гравця для гри в бліц-опитування в першому раунді використовуючи клієнтську частину	Серверний застосунок зберіг у пам’яті інформацію про гравця, який буде відповідачем у раунді, та записав у лог подію вибору відповідача	Пройдено
2	Відповідальні гравці відповідають на бліц-опитування використовуючи клієнтську частину	Серверний застосунок записав у пам’яті інформацію про раунд, та записав у лог його події	Пройдено

Кінець таблиці Д.3

Вибір відповідача у бліц-опитуванні в першому раунді гри Brain Knights			
3	Капітан команди знову обирає того самого гравця для наступного раунду в бліц-опитуванні, використовуючи клієнтську частину	Серверний застосунок перевіряє отримані дані обраного гравця з даними масиву вже відповівших гравців. Гравець не записується відповідальним у раунді	Пройдено
Результати тестування			
Тестувальник: Калініченко О. Ю.	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»

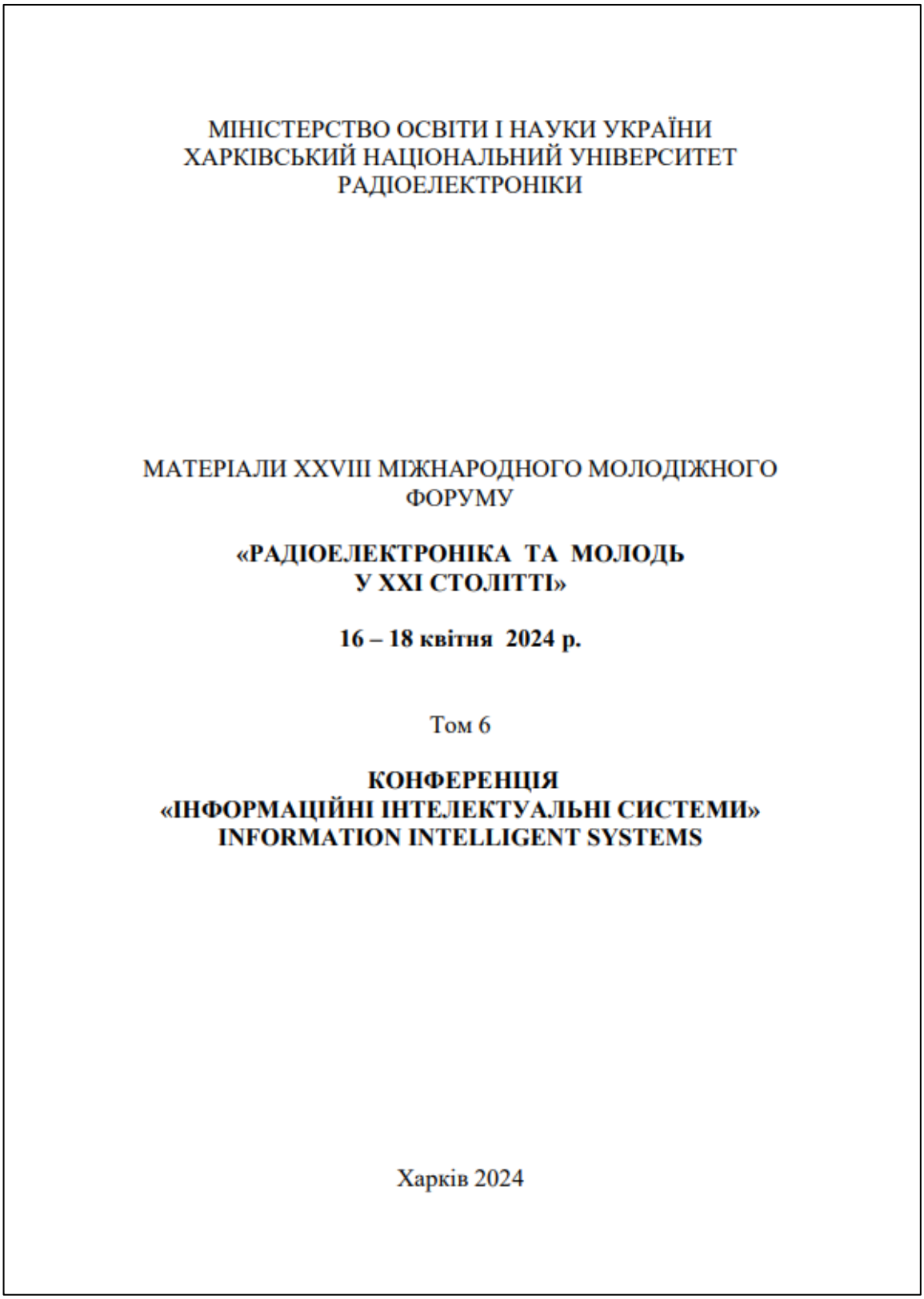
Голова оргкомітету Форуму Ігор РУБАН

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)

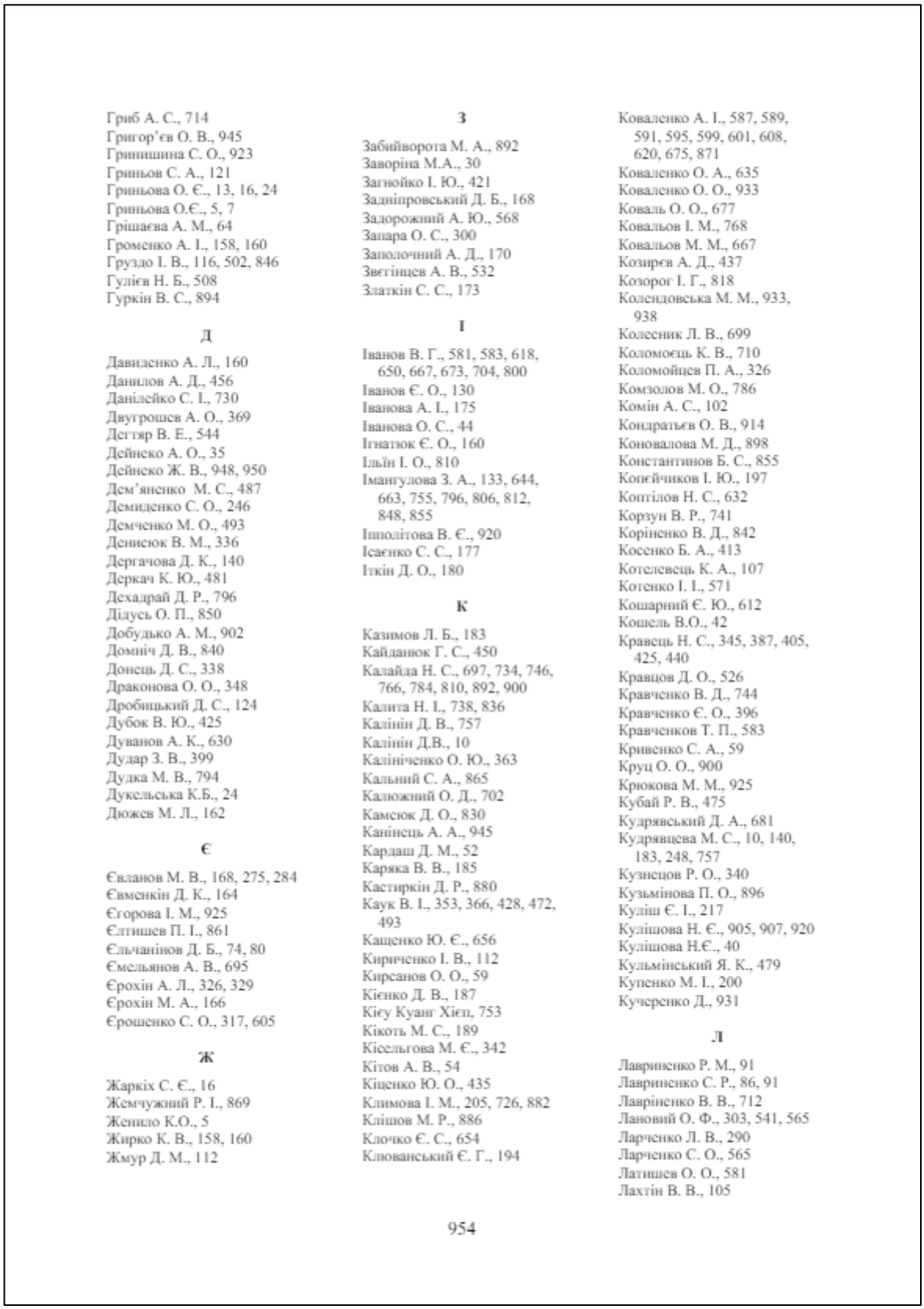


Рисунок И.2 – Алфавітний покажчик (знімок екрана виконано самостійно)

УДК 004.514:004.89 DOI: https://doi.org/10.30837/IYF.IIS.2024.363

ІМІТАЦІЯ РЕАЛІСТИЧНОГО ІГРОВОГО СУПРОТИВНИКА ЗА ДОПОМОГОЮ ШТУЧНОГО ІНТЕЛЕКТУ НА ОСНОВІ МОДЕЛІ GPT TEXT-DAVINCI-003

Калініченко О. Ю.
Науковий керівник – ст. викл. Новіков Ю. С.
Харківський національний університет радіоелектроніки, каф. ПІ
м. Харків, Україна
e-mail: oleksandr.kalinichenko1@nure.ua

This thesis investigates the application of artificial intelligence to model a realistic game opponent. The main focus is on analyzing a variety of artificial intelligence approaches and models in order to develop strategies for creating the most realistic bot behavior in the game space. In addition, the paper pays attention to the importance of proper selection of cues, which play a key role in ensuring interactivity and dynamics of the gameplay. The results of the study reveal strategic approaches to creating an adversary that can effectively adapt to different game scenarios and provide players with an exciting and unpredictable gaming experience.

Впровадження штучного інтелекту (ШІ) в ігровий процес є ключовою складовою сучасної розважальної індустрії. Це відкриває перед розробниками можливості створення більш реалістичних та захоплюючих ігрових світів. Постійне вдосконалення застосовуваного ШІ в іграх свідчить про його значущість і потенціал для оптимізації процесу розробки, створення сценаріїв, живого оточення, а також створення реалістичних поведінкових моделей.

Існує кілька методів створення або інтеграції штучного інтелекту в іграх, включаючи алгоритми розв'язання задач, машинне навчання, генетичні алгоритми та нейронні мережі. Вибір ШІ GPT для власної гри обґрунтовується його здатністю гнучко адаптуватися до різних питань та ситуацій, а також широкими знаннями, що базуються на великому об'ємі інформації, зібраної з Інтернету. Крім того, його використання економічно та технічно ефективне, оскільки не потребує значних інвестицій у розробку спеціалізованих алгоритмів чи навчання складних моделей. Text-Davinci-003 від компанії OpenAI ідеально підходить для інтеграції у гру, оскільки він використовується у віртуальних помічниках, чат-ботах служби підтримки та пошукових системах, що базуються на обробці природної мови. Також варто зазначити, що обрана модель GPT має важливу характеристику – здатність пояснювати прийняті рішення. Пояснюваність – це набута властивість процесу прийняття рішень, яка зазвичай реалізується за допомогою зовнішніх засобів [1]. Це надає можливість отримувати інформацію щодо того, як саме модель приймає свої рішення, що дозволяє краще налаштовувати її та покращувати ігровий досвід гравців.

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

Ефективне написання промптів є важливою частиною процесу створення реалістичного ігрового супротивника. Промпт – це вхідні дані, надані моделі ШІ, які встановлюють контекст, ціль або обмеження для відповіді моделі [2]. Важливо врахувати кілька ключових аспектів, щоб забезпечити якість та реалізм взаємодії гравців з ботом:

– промпт повинен бути адаптивним до контексту гри та дій гравців;

– промпт повинен бути гнучким та здатним реагувати на нестандартні або неочікувані дії гравців;

– промпт повинен мати чітко сформульований запит;

– промпт повинен мати обмеження з точки зору кількості символів або слів, щоб уникнути надмірно довгих відповідей;

– промпт повинен бути налаштований на надання не тільки правильних, а й некоректних відповідей для імітації людської поведінки.

Реалізація процесу включення штучного інтелекту в ігровий процес передбачає передачу даних користувачів зі сторони клієнта на сервер, включаючи поставлене запитання та варіанти відповідей усіх гравців на нього. Система спочатку ставить перед ШІ завдання «замаскуватися» під людину, генеруючи відповідь, схожу на відповіді інших гравців. Зазначена відповідь формується з урахуванням вказівок, що обмежують кількість символів та передбачають варіанти відповідей залежно від інших гравців. У випадку, якщо всі гравці дали неправильну відповідь, ШІ також повинен надати неправильну відповідь. Але якщо принаймні одна з відповідей була правильною, він також повинен надати правильну відповідь, що буде схожа на неї. Після цього запит відправляється для обробки за допомогою інтерфейсу програмування застосунків (API), наданого компанією OpenAI. Отримана відповідь додається до загального результату всіх відповідей та повертається на клієнт. У випадку, якщо виникла помилка під час використання API OpenAI, система генерує випадкову загальновживану відповідь з попередньо підготовленого списку, що зберігається на сервері. Опис процесу інтеграції та налаштування ШІ в системі зображено на рисунку 1 та 2.

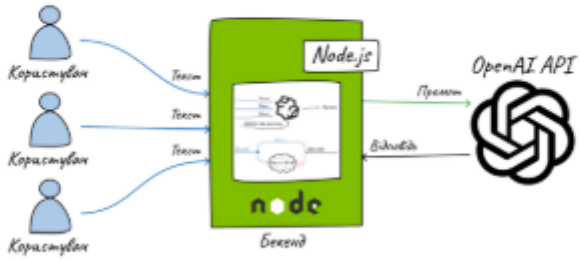


Рисунок 1 – Схема інтеграції штучного інтелекту в систему

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

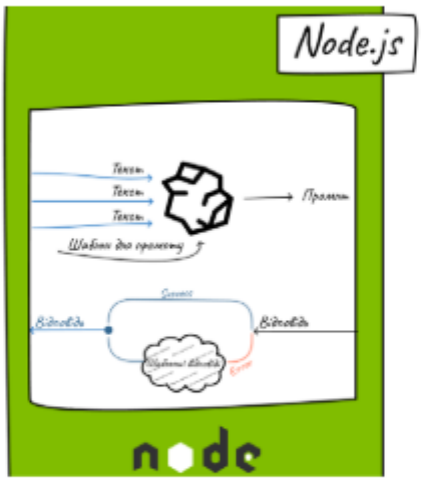


Рисунок 2 – Схема налаштування ШІ в системі

Таким чином, процес інтеграції штучного інтелекту на основі моделі GPT Text-Davinci-003 у ігровий процес виявляється перспективною та ефективною стратегією для створення реалістичного ігрового супротивника. Використання цієї моделі дозволяє створити бота, який демонструє свою гнучкість у спілкуванні та адаптацію до різних ситуацій у грі. Процес створення промптів для взаємодії з ботом потребує уважного аналізу та налаштування, щоб забезпечити реалістичність та якість гри. Правильно налаштований ігровий супротивник забезпечить гравцям цікаві та непередбачувані ігрові ситуації, що покращує ігровий досвід користувачів.

Список використаних джерел:

1. Chalyi S., & Leshchynskyi, V. (2023). Possible evaluation of the correctness of explanations to the end user in an artificial intelligence system. Advanced Information Systems, 7(4), 75–79. https://doi.org/10.20998/2522-9052.2023.4.10

2. Mayo, Matthew. Mastering Generative AI and Prompt Engineering: A Practical Guide for Data Scientists / Matthew Mayo, KDnuggets Editor-in-Chief. – Data Science Horizons, 2023. – 41.

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)