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, налаштування мережевої гри
(тема)

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

Бухало В. О.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Програмна інженерія
(повна назва освітньої програми)

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

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

2024 р.

Харківський національний університет радіоелектроніки

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

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Бухало Володимиру Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи Ігровий програмний застосунок у жанрі Multiplayer Party Games. Back-end, налаштування мережевої гри
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 8 квітня 2024р.

Студент (ка)_____ Бухало В. О.
(підпис)

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

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, 194 стор., 50 рис., 2 табл., 12 джерел, 8 додатків.

ІГРОВИЙ ПРОГРАМНИЙ ЗАСТОСУНОК, МЕРЕЖЕВА ГРА, УПРАВЛІННЯ ФАЙЛАМИ, BACK-END, FRONT-END, MULTIPLAYER PARTY GAMES, NESTJS, NODE.JS, SOCKET.IO, TYPESCRIPT, WEBSOCKET

Об’єкт розробки – back-end та налаштування мережевої гри в ігровому програмному застосунку у жанрі multiplayer party games.

Мета розробки – створення back-end частини системи та налаштування мережевої гри в ігровому програмному застосунку жанру multiplayer party games.

Метод рішення – середовище розробки Visual Studio Code, програмна платформа Node.js, фреймворк NestJS, WebSocket-сервер, бібліотека Socket.IO та мови програмування TypeScript і JavaScript.

У результаті розробки була створена back-end частина системи та налаштована мережева гра в ігровому програмному застосунку у жанрі multiplayer party games, який містить три гри. У межах кваліфікаційної роботи розроблена back-end частина системи охоплює логіку двох багатокористувацьких ігор для вечірок наступних жанрів: соціальна дедуктивна гра та гумористична карткова.

GAME SOFTWARE APPLICATION, NETWORK GAME, FILE MANAGEMENT, BACK-END, FRONT-END, MULTIPLAYER PARTY GAMES, NESTJS, NODE.JS, SOCKET.IO, TYPESCRIPT, WEBSOCKET

The object of development is the back-end part of the system and the configuration of a network game in a multiplayer party games software application.

The purpose of development is to create the back-end part of the system and set up a network game in a multiplayer party games software application.

The solution method used was the Visual Studio Code development environment, Node.js software platform, NestJS framework, WebSocket server, Socket.IO library, and TypeScript and JavaScript programming languages.

As a result of the development, the back-end part of the system was created and a network game was set up in a multiplayer party games software application containing three games. Within the framework of the qualification work, the developed back-end part of the system covers the logic of two multiplayer party games of the following genres: a social deduction game and a humorous card game.

Я, Бухало Володимир Олександрович, 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, налаштування мережевої гри», 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

PC – Personal Computer

ШІ – Штучний Інтелект

HTTP – Hypertext Transfer Protocol

UML – Unified Modeling Language

ПЗ – Програмне Забезпечення

API – Application Programming Interface

OpenAI – Open Artificial Intelligence

ChatGPT – Chat Generative Pre-trained Transformer

CORS – Cross-Origin Resource Sharing

ВСТУП

З інтенсивним поширенням доступу до Інтернету, ряд аспектів соціальної взаємодії, які раніше вирішувалися шляхом фізичної присутності в одному місці, тепер можуть здійснюватися через цифрові засоби зв’язку. Це явище призвело до збільшення кількості осіб, які можуть спілкуватися, грати в ігри та підтримувати дружні стосунки, незважаючи на велику відстань між ними. Крім того, поширення доступу до Інтернету сприяло популяризації багатокористувацьких відеоігор. Наприклад, на платформі Steam на початку квітня 2024 року саме багатокористувацькі відеоігри займають провідні позиції за кількістю одночасних гравців. Також, зростання користувачів Інтернету разом із стрімким розвитком інформаційних технологій створює можливість перенесення деяких жанрів ігор, таких як ігри для вечірок, у цифровий формат. У такому контексті розробка ігрового програмного застосунку у жанрі multiplayer party games набуває особливої актуальності, оскільки це створює нові можливості для розваг під час особистих або віртуальних зустрічей між друзями та знайомими. Внаслідок цього, досвід розробки такого програмного забезпечення, його back-end частини і налаштування мережевої гри також стають актуальними.

Темою кваліфікаційної роботи є «Ігровий програмний застосунок у жанрі Multiplayer Party Games. Back-end, налаштування мережевої гри». Основне завдання ігрового програмного застосунку у жанрі multiplayer party games полягає у наданні можливості множині користувачів грати в надані ігри.

Back-end частина системи відповідає за керування грою, включаючи створення кімнат для гравців, обробку їхніх дій та реагування на їх запити під час гри, синхронізацію гри між усіма гравцями, а також за обробку та збереження даних сесії гри протягом часу їх існування.

Налаштування мережевої гри відповідає за створення WebSocket-серверу та його налаштування з використанням бібліотеки Socket.IO на back-end частині, а також за інтеграцію Socket.IO для налаштування WebSocket-з’єднання на

клієнтській стороні з метою забезпечення успішної комунікації між клієнтами та сервером. Це дозволить клієнтській стороні використовувати весь функціонал мережевої гри, який надає back-end частина.

Метою роботи є розробка back-end частини системи та налаштування мережевої гри в ігровому програмному застосунку жанру multiplayer party games, який містить три гри. Кожна з них буде багатокористувацькою грою для вечірок і буде належати до одного з наступних жанрів: соціальна дедуктивна гра, гумористична карткова гра та змагальна вікторина. Ця робота охоплюватиме розробку логіки двох з них – соціальної дедуктивної гри та гумористичної карткової гри. Для розробки використовуватимуться середовище розробки Visual Studio Code, програмна платформа Node.js, фреймворк NestJS, WebSocket, бібліотека Socket.IO та мови програмування TypeScript і JavaScript.

Ігровий програмний застосунок у жанрі multiplayer party games може бути використаний для розваг під час особистих або віртуальних зустрічей між знайомими та друзями. Для організації ігрового процесу необхідний лише доступ до Інтернету, веб-браузер, персональний комп’ютер з монітором для створення ігрової сесії та відображення даних хоста, а також смартфони для гравців, які будуть підключатися до ігрової сесії. Програмний застосунок повинен бути привабливим через свою простоту та доступність, адже для початку гри немає необхідності встановлювати нічого, окрім веб-браузера. Це робить його гарним вибором для широкого кола аудиторії, від дорослих до підлітків, а також для різних заходів, від онлайн-вечірок з друзями до особистих зустрічей.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Ігровим програмним застосунком є програмне забезпечення, створене для розваг, яке надає користувачам доступ до віртуального середовища та дозволяє взаємодіяти з ним через різні пристрої введення, такі як клавіатура, миша або сенсорний екран.

Жанр ігор party games зазвичай характеризує ігри, які призначені для групових інтерактивних сесій, де гравці можуть взаємодіяти один з одним. Головна мета таких ігор – розважити гравців та сприяти соціальній взаємодії. Вони чудово підходять для будь-яких вечірок, зустрічей з друзями або навіть корпоративних заходів, де група людей хоче відпочити та весело провести час разом.

Ігрові програмні застосунки жанру multiplayer party games призначені для розваг під час неформальних особистих або віртуальних зустрічей між друзями та знайомими. Зазвичай вони включають в себе серію простих відеоігор, спрямованих на соціальну взаємодію та розваги. Часто ці відеоігри можуть бути різноманітними за жанром та стилем гри, такими як соціальні дедуктивні ігри, гумористичні карткові ігри та змагальні вікторини.

Цим відеоіграм зазвичай притаманні наступні характеристики:

– багатокористувацька гра: у центрі цього жанру лежить ідея взаємодії між гравцями, тому вони надають можливість грати одночасно кільком людям на одному екрані або через мережу;

– прості правила: щоб всі могли швидко приєднатися і зрозуміти, що відбувається, multiplayer party games зазвичай мають прості та інтуїтивно зрозумілі правила; це дозволяє гравцям з різним рівнем досвіду грати разом без особливих перешкод;

– розважальний контент: ігри цього жанру зазвичай орієнтовані на розваги і створення атмосфери веселощів; вони можуть включати різні завдання або головоломки, які викликають сміх і конкуренцію між гравцями;

– графіка та звук: важливою частиною multiplayer party games є візуальне та звукове оформлення, яке зазвичай яскраве, кольорове та веселе; музика, звукові ефекти та анімація використовуються для підтримки атмосфери веселощів та розваг;

– змагальний аспект: часто в іграх цього жанру присутній елемент змагання між гравцями, який може бути як безпосереднім, так і опосередкованим; це може бути змагання в виконанні завдань, боротьба за рейтинги або просто боротьба за перемогу.

Також, деякі ігри жанру multiplayer party games мають розподіл на ролі організатора та гравця всередині всього ігрового програмного застосунку незалежно від обраної гри.

Організатор відповідає за проведення гри та має ряд можливостей для налаштування та контролю загального потоку гри. Якщо гра відбувається під час особистої зустрічі гравців, які знаходяться в одній кімнаті, то екран організатора демонструється на великому екрані для відображення потоку гри. Якщо гра відбувається під час віртуальної зустрічі, то демонстрація зазвичай відбувається за допомогою вбудованих інструментів у додатку, в якому відбувається зустріч.

Гравець приєднується до гри та бере участь у конкурсах чи інших активностях, які пропонує гра. Він бере активну участь у грі та впливає на її хід, виконуючи різноманітні завдання та виклики, що запропоновані в грі. Гравці можуть бути поділені на інші ролі, які пропонує гра, до якої вони приєдналися.

Зазвичай організатори та гравці мають свою власну різну клієнтську частину, яка забезпечує доступ до необхідного для них функціоналу та візуального супроводження. Організатори використовують клієнтську частину, що надає функціонал для ініціалізації ігрової сесії в якості хоста, а гравці користуються контролерами, якими можуть бути смартфони або інші пристрої, що дозволяють підключатися до ініціалізованої ігрової сесії та взаємодіяти з грою та виконувати дії, передбачені особливостями геймплею.

Оскільки ігри цього жанру є багатокористувацькими, для того, щоб забезпечити спілкування та взаємодію між усіма гравцями під час гри, вони

зазвичай мають back-end частину. Її суть полягає у керуванні всією мережевою інфраструктурою гри. Вона відповідає за обробку даних, переданих між гравцями, синхронізацію дій у реальному часі, а також збереження та обробку результатів гри. Це означає, що back-end частина гри відповідає за багато аспектів, таких як лобі гри, підключення гравців, управління сеансами гри та інші важливі аспекти, що дозволяють багатокористувацьким іграм працювати без перешкод.

Також у іграх цього жанру, де гравці грають разом з великими групами друзів, ефективна та надійна мережева інфраструктура є ключовою для успішної гри. Вона забезпечує плавну та зручну гру для усіх гравців, дозволяючи взаємодіяти з ігровим середовищем та іншими гравцями у реальному часі.

Щодо актуальності ігрових програмних застосунків жанру multiplayer party games можна констатувати, що їх соціальний аспект, їх здатність об’єднувати гравців у веселі групові заняття, робить їх особливо привабливими в епоху віртуальних спілкувань та віддалених форм взаємодії. Крім того, зростаюча доступність Інтернету і розвиток мобільних технологій дозволяють гравцям легко з’єднуватися з іншими у будь-якому місці і в будь-який час. Тому, без сумніву, такі ключові фактори, як простота, доступність і соціальний аспект, мають свою аудиторію на ринку і можуть сприяти росту популярності даного жанру. У свою чергу, ризики, які можуть призвести до зменшення популярності ігрових програмних застосунків цього жанру, можуть бути пов’язані з деякою втомою від стандартних механік гри або конкуренцією з іншими жанрами, які тимчасово стають більш привабливими для гравців. Також варто враховувати, що мода в ігровій індустрії може швидко змінюватися, а отже, популярність будь-якого жанру є динамічним і непостійним явищем.

Цільова аудиторія ігор у жанрі multiplayer party games складається з широкого спектру гравців, від підлітків до дорослих будь-якого віку. Ці ігри приваблюють людей, які цінують соціальні взаємодії та забаву у компанії друзів чи родичів. Вони надають можливість не лише активно взаємодіяти з іншими учасниками гри, а й відчувати атмосферу веселощів та конкуренції. Часто це групи людей, які збираються разом на вечірку, відпочинок у кафе чи просто онлайн, щоб

провести час разом та насолодитися неповторними моментами, які створюються завдяки ігровому процесу. Таким чином, ця аудиторія характеризується соціальністю, відкритістю до нових вражень та бажанням поділитися радістю гри з іншими.

На сучасному ринку ігрової індустрії спостерігається різноманітність ігрових програмних застосунків у жанрі мультиплеєрних вечірок, наприклад The Jackbox Party Pack, який містить уже 10 частин, або Gartic Phone. Усі вони існують не один рік і заслужили довіру аудиторії цього жанру ігор. Все це має значний вплив на рівень конкуренції. Різноманітність цих ігор виявляється у різних стилях, геймплеях, технічних рішеннях та механіках, що створює широкі можливості для вибору для гравців. З цієї причини розробники намагаються створювати нові рішення, оскільки кожен з них намагається привернути увагу гравців до свого продукту. Для досягнення конкурентоспроможності в такому середовищі важливо постійно підтримувати свій продукт. Це може включати постійне оновлення гри, додавання нового контенту, вдосконалення геймплею та врахування актуальних тенденцій у світі геймінгу. Крім того, ключовим аспектом успіху є здатність привертати увагу гравців своєю оригінальністю та якістю геймплею.

Метою ігрових програмних застосунків у жанрі multiplayer party games є створення веселої та соціально зближувальну атмосфери для гравців розважальних відеоігор, надаючи захоплюючий ігровий досвід під час групових інтерактивних сесій. Їх конкретними цілями є:

– створення веселої та розважальної атмосфери, щоб гравці могли приємно провести час разом з друзями або знайомими;

– сприяння соціальній взаємодії між гравцями, стимулюючи комунікацію та спільні активності під час гри;

– забезпечення простоти та доступності застосунку, щоб гравці швидко могли приєднатися до гри та розуміти її правила;

– стимулювання змагального духу, надаючи гравцям можливість конкурувати між собою у відповідних викликах або завданнях;

– забезпечення стабільної роботи та мережевої взаємодії, щоб дії гравців синхронізувалися в реальному часі, лобі гри працювало коректно, гравці могли без перешкод підключатися до нього, а також забезпечення ефективного управління сеансами гри та іншими важливими аспектами для безперебійної роботи.

Визначимо фрагменти предметної області з роботи ігрових програмних застосунків у жанрі multiplayer party games та для кожного фрагмента предметної області виділимо: об’єкти, процеси та користувачів; інформаційні потреби та результати діяльності кожної групи користувачів або підрозділів; загальні характеристики процесів споживання та обробки інформації. Фрагменти предметної області:

а) графічне та аудіовізуальне оформлення:

1) об’єкти: графіка, анімація, звукові ефекти;

2) процеси: відтворення графічних та звукових ефектів, анімацій;

3) користувачі: гравці та організатори гри;

4) інформаційні потреби та результати: гравці та організатори очікують від гри якісну графіку, яка створює атмосферу та настрій, а також звуковий супровід;

5) загальні характеристики процесів споживання та обробки інформації: графіка та звук обробляються та відтворюються на пристроях гравців, забезпечуючи їм візуальне та аудіо сприйняття гри;

б) механіки гри та правила:

1) об’єкти: геймплейні елементи, правила гри;

2) процеси: взаємодія гравців з ігровими об’єктами, визначення результатів дій гравців;

3) користувачі: гравці;

4) інформаційні потреби та результати: гравці очікують чітких правил гри та можливості взаємодії з ігровими об’єктами відповідно до цих правил;

5) загальні характеристики процесів споживання та обробки інформації: правила гри та механіки обробляються грою для визначення результатів дій гравців та управління ходом гри;

в) підтримка різних платформ та веб-браузерів:

1) об’єкти: персональні комп’ютери, мобільні пристрої, веб-браузери;

2) процеси: адаптація гри для різних пристроїв;

3) користувачі: гравці та організатори гри;

4) інформаційні потреби та результати: гравці та організатори очікують правильну роботу ігрового програмного застосунку на своєму пристрої;

5) загальні характеристики процесів споживання та обробки інформації: гра повинна бути адаптована для різних платформ та веб-браузерів, щоб забезпечувати правильне відображення ігрового процесу;

г) інтерфейс користувача:

1) об’єкти: елементи інтерфейсу, такі як меню, кнопки, налаштування;

2) процеси: відображення інформації, обробка дій користувачів;

3) користувачі: гравці та організатори гри;

4) інформаційні потреби та результати: гравці та організатори гри потребують, щоб інтерфейс гри був зручним та дозволяв взаємодіяти з усіма необхідними функціями гри;

5) загальні характеристики процесів споживання та обробки інформації: інтерфейс гри забезпечує візуалізацію та взаємодію з ігровими елементами для користувачів;

д) мережева взаємодія:

1) об’єкти: клієнтські пристрої, сервер;

2) процеси: передача даних, синхронізація дій гравців, обробка запитів;

3) користувачі: гравці та організатори гри;

4) інформаційні потреби та результати: гравці та організатори гри очікують можливість приєднання до ігрових сесій та відображення дій інших гравців в них;

5) загальні характеристики процесів споживання та обробки інформації: збереження та передача даних в мережі в реальному часі, обробка даних для забезпечення синхронізації гри.

Для більш повного розуміння проблеми та кращого розуміння інтерфейсу та функціоналу майбутнього ігрового програмного застосунку у жанрі multiplayer party games, розглянемо існуючі аналоги: The Jackbox Party Pack та Gartic Phone.

The Jackbox Party Pack – набір ігор, розроблений студією Jackbox Games та доступний на різних платформах, таких як ПК, консолі та мобільні пристрої. Кожна гра включена в набір, це унікальний мультиплеєр, де гравці можуть приєднатися за допомогою своїх телефонів або комп'ютерів і брати участь у різних вікторинах, тестах і творчих завданнях.

Після запуску встановленого The Jackbox Party Pack на своєму пристрої організатор натискає кнопку старту, і перед ним відкривається екран, де наведено список ігор (рис. 1.1), які доступні в цьому застосунку. Тут він може обрати одну з них та перейти до неї. Крім списку, на екрані присутні елементи навігації, налаштування, пояснення та перемикач для ввімкнення або вимкнення сімейного режиму.

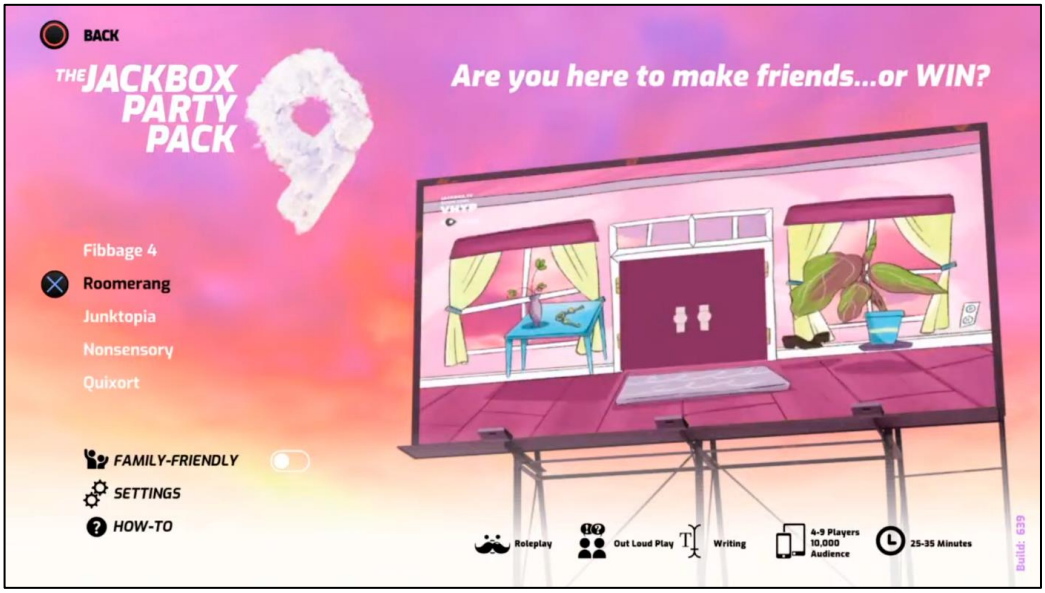


Рисунок 1.1 – Екран зі списком доступних ігор в The Jackbox Party Pack 9 (знімок екрана виконано самостійно)

Після того, як було обрано гру зі списку, відкривається екран головного меню цієї гри, де відображено її назву, а також наявні елементи навігації, налаштування та початку гри (рис. 1.2).

Рисунок 1.2 – Екран головного меню гри в The Jackbox Party Pack 9 (знімок екрана виконано самостійно)

Далі відкривається екран з кімнатою гри (рис. 1.3). Коли в кімнаті буде присутня мінімальна кількість гравців, то володар кімнати зможе через свій пристрій запустити гру.

Рисунок 1.3 – Екран кімнати гри в The Jackbox Party Pack 9 (знімок екрана виконано самостійно)

Через свої пристрої гравці відкривають веб-браузер та переходять на веб-сайт https://jackbox.tv/, де вони можуть підключитися до кімнати, після введення правильного коду кімнати, свого імені та натиснувши кнопку «Play» (рис. 1.4).

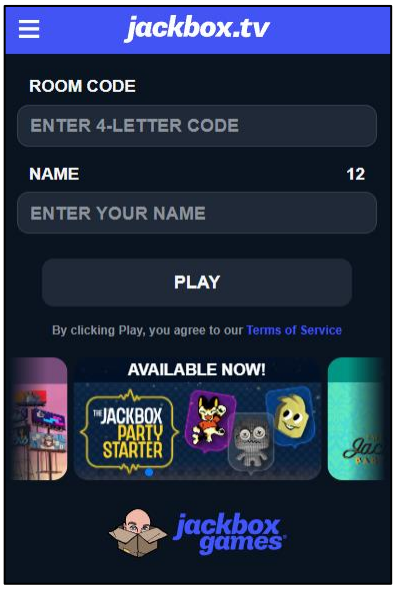


Рисунок 1.4 – Екран для підключення до кімнати через веб-сайт https://jackbox.tv (знімок екрана виконано самостійно)

Після того, як гра була розпочата, гравців ознайомлюють з правилами гри через короткий відеоролик (рис. 1.5).

Рисунок 1.5 – Екран пояснення правил гри в The Jackbox Party Pack 9 (знімок екрана виконано самостійно)

Через свої пристрої гравці, після того, як вже знаходяться в розпочатій грі, можуть виконувати потрібні дії згідно з правилами гри (рис. 1.6).

Рисунок 1.6 – Екран з відображенням ігрового процесу на пристрої гравця (знімок екрана виконано самостійно)

Потік гри відображається на екрані організатора (рис. 1.7). Гравці можуть бачити свої дії та їх результати на цьому екрані.

Рисунок 1.7 – Екран з відображенням ігрового процесу розпочатої гри в The Jackbox Party Pack 9 (знімок екрана виконано самостійно)

Під кінець гри відображаються всі гравці та їх кінцеві бали, які вони отримали (рис. 1.8).

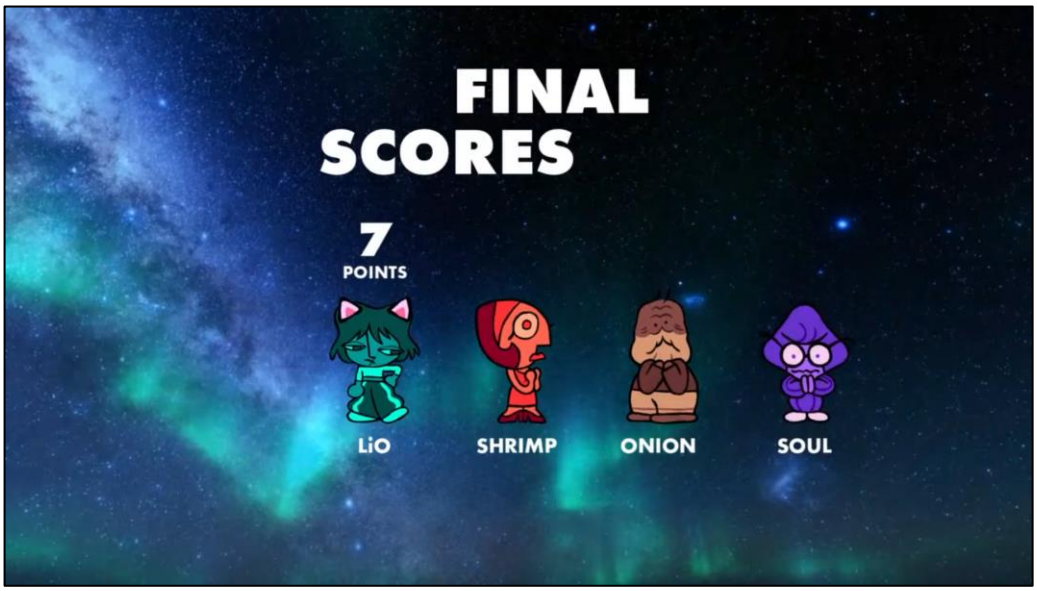


Рисунок 1.8 – Екран з отриманими балами гравцями під час гри в The Jackbox Party Pack 9 (знімок екрана виконано самостійно)

Також наводиться переможець, якого було обрано відповідно до отриманої кількості балів (рис. 1.9).

Рисунок 1.9 – Екран переможця гри в The Jackbox Party Pack 9 (знімок екрана виконано самостійно)

Розглянемо наступний аналог. Gartic Phone, розроблена студією Onrizon Social Games. В цій грі кожен учасник робить свій внесок у створення історії або малюнка, що призводить до веселих, унікальних та несподіваних ігрових історій.

Після відкриття веб-додатку https://garticphone.com/, користувачі потрапляють на екран головного меню (рис. 1.10). Тут можна переглянути інформацію про правила гри, соціальні мережі, налаштувати мову, ввести своє ім'я, обрати аватар та розпочати ігрову сесію, натиснувши кнопку «Start». Можна увійти як анонімний користувач або авторизуватися у системі через Discord або Twitch.

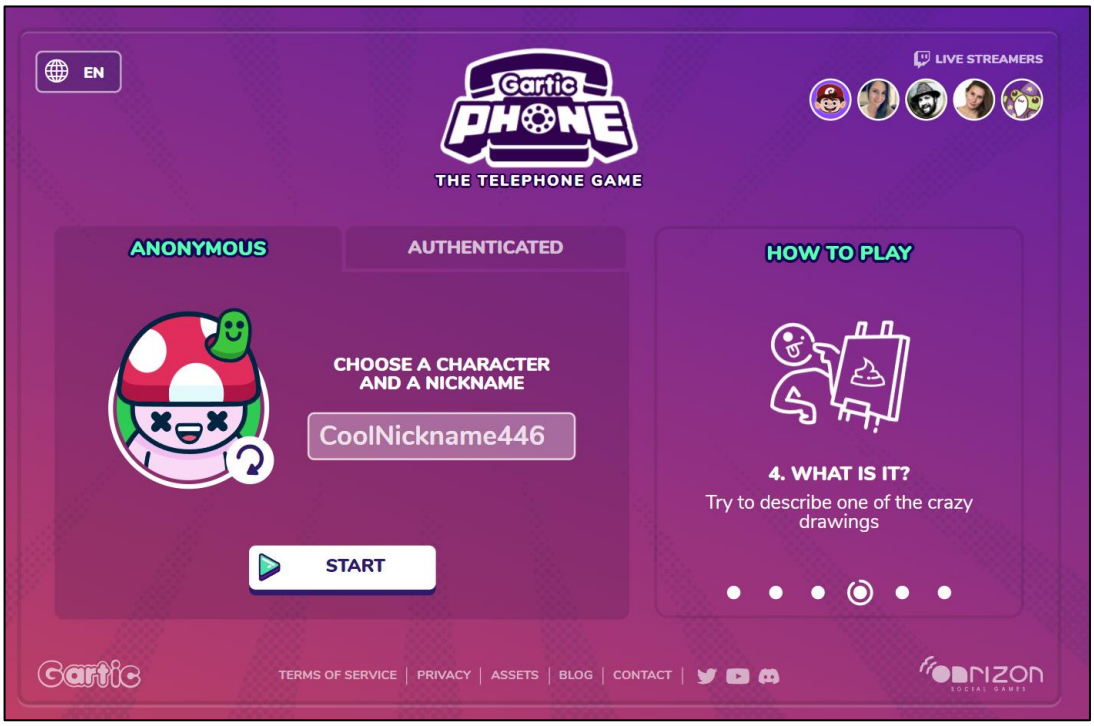


Рисунок 1.10 – Екран головного меню гри Gartic Phone на веб-сайті https://garticphone.com/ (знімок екрана виконано самостійно)

Після створення ігрової сесії, гравець переходить на екран налаштування кімнати (рис. 1.11), де він може вказати кількість гравців, обрати один із запропонованих режимів гри, скопіювати посилання за допомогою кнопки «Invite», щоб інші гравці могли приєднатися до кімнати, а також розпочати гру, натиснувши кнопку «Start».

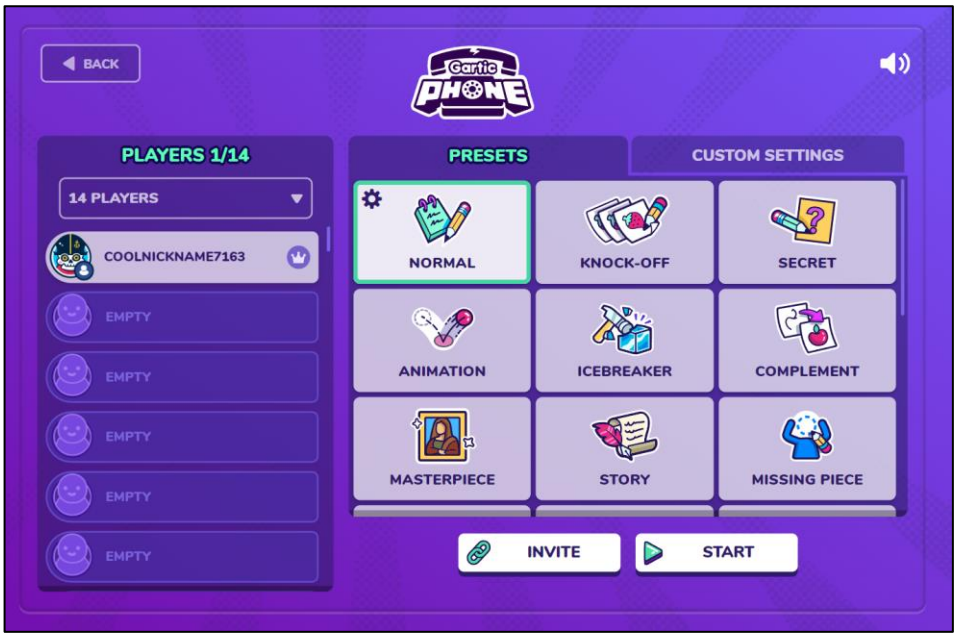


Рисунок 1.11 – Екран створеної кімнати гри Gartic Phone на веб-сайті https://garticphone.com/ (знімок екрана виконано самостійно)

В залежності від обраного режиму користувачі бачать перед собою екран, на якому вказане поточне завдання (рис. 1.12), а також наведені приклади його виконання або надані всі необхідні інструменти для його виконання. Суть всіх режимів можна поділити на малювання або написання історій.

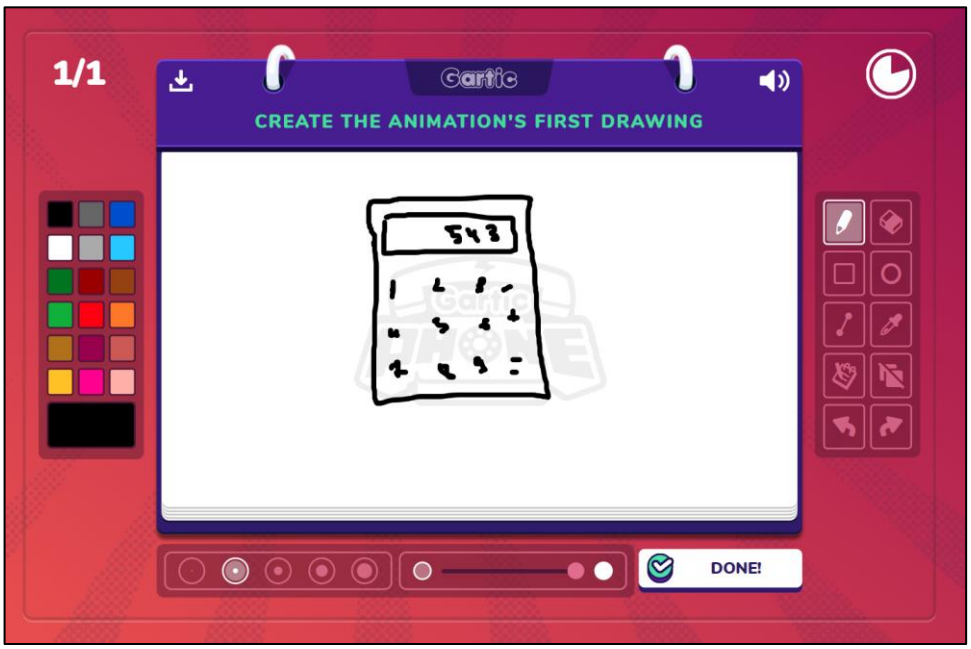


Рисунок 1.12 – Екран завдання гри Gartic Phone на веб-сайті https://garticphone.com/ (знімок екрана виконано самостійно)

Після виконання всіма гравцями завдань наступає етап підсумків (рис. 1.13), на якому представляються результати гри: кожен гравець має свій альбом, у якому відображено, як кожен учасник вплинув на кінцевий результат або власну створену історію, яку збагачували всі учасники гри.

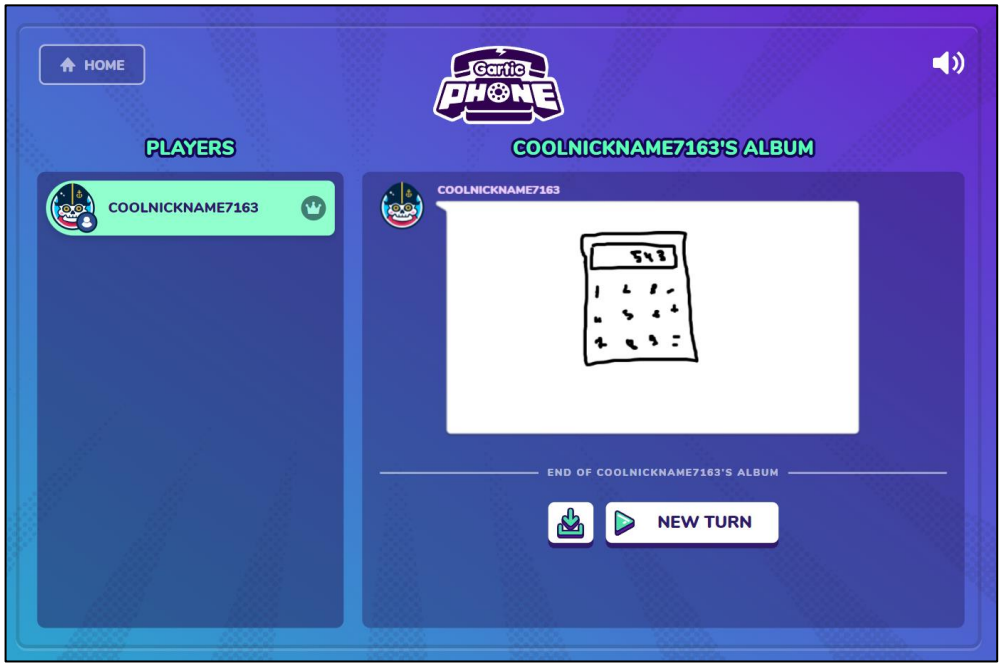


Рисунок 1.13 – Екран підсумку гри Gartic Phone на веб-сайті https://garticphone.com/ (знімок екрана виконано самостійно)

Проведемо аналіз розглянутих аналогів визначивши її функціональне призначення та переваги й недоліки.

Функціональне призначення обох ігор полягає в наданні середовища для розваг.

Переваги Gartic Phone:

– простота використання: Gartic Phone – це веб-застосунок, тому для гри потрібно лише браузер і доступ до Інтернету;

– легко долучитися: гравцям не потрібно завантажувати або встановлювати додаткові програми або ігрові платформи;

– безкоштовно: базова версія Gartic Phone є безкоштовною, що робить її доступною для всіх;

– гра в режимі реального часу: гравці можуть грати одночасно, надсилаючи та переглядаючи малюнки та описи миттєво.

До недоліків Gartic Phone можна віднести те, що, незважаючи на те, що ця гра пропонує кілька режимів гри, вона може швидко стати одноманітною порівняно з іншими іграми, оскільки в основному всі режими можуть бути зведені до малювання або написання історій.

Переваги The Jackbox Party Pack:

– різноманітність ігор: кожен The Jackbox Party Pack містить кілька різних ігор, що дозволяє гравцям обирати та спробувати різноманітні варіанти розваг;

– кросс-платформеність: багато ігор The Jackbox Party Pack доступні на різних платформах, таких як PC, консолі та мобільні пристрої;

– інтерактивність: багато ігор в Jackbox використовують смартфони гравців як контролери, що робить гру більш інтерактивною та веселою.

Недоліки The Jackbox Party Pack:

– вартість: кожен The Jackbox Party Pack коштує гроші, що може бути перешкодою для тих, хто не хоче витрачати гроші на ігри;

– залежність від групи: деякі ігри The Jackbox Party Pack краще грати з великою групою людей, тому вони можуть бути менш привабливими для використання в малій компанії або з невеликою сім’єю.

Після аналізу існуючих аналогів можна зробити висновок, що для того, щоб покращити конкурентоспроможність майбутнього ігрового програмного застосунку, важливо намагатися об’єднати переваги всіх розглянутих прикладів та виправити їх недоліки.

1.2 Виявлення та вирішення проблем

З інтенсивним поширенням доступу до Інтернету, ряд аспектів соціальної взаємодії, які раніше вирішувалися шляхом фізичної присутності в одному місці,

тепер можуть здійснюватися через цифрові засоби зв'язку. Це явище призвело до збільшення кількості осіб, які можуть спілкуватися та грати в ігри, незважаючи на велику відстань між ними. Що в свою чергу створює попит на багатокористувацькі ігри, орієнтовані на соціальну взаємодію. У цих іграх такі особи можуть проводити свій вільний час, спілкуючись один з одним та розважаючись. Одним з жанрів багатокористувацьких ігор, геймплей яких орієнтований на соціальну взаємодію, є multiplayer party games. У цих іграх часто грають компанії друзів або родичів, щоб розважитися та створити зближувальну атмосферу.

Отже, для задоволення попиту на ігри з якостями, що характерні для жанру multiplayer party games, гарним рішенням буде створення ігрового програмного застосунку саме у цьому жанрі. Щоб забезпечити простий доступ, можна розмістити цей ігровий застосунок у веб-браузері. Таким чином, його не потрібно буде завантажувати, щоб грати. Достатньо мати лише пристрій з веб-браузером та доступом до Інтернету. Для подолання бар’єру входу в гру, ігри для вечірок у цьому застосунку повинні мати прості та інтуїтивно зрозумілі правила, щоб всі гравці могли швидко приєднатися і зрозуміти, що необхідно робити. Щодо підтримки інтересу гравців, ігровий застосунок повинен включати різноманітні ігри для вечірок з різним геймплеєм, щоб привернути увагу гравців. Потрібно також стежити за ринком і забезпечувати постійні оновлення та підтримку продукту, щоб зберегти зацікавленість гравців і забезпечити актуальність на ринку.

Загалом, своєчасне виявлення та вирішення усіх проблем дозволить підтримувати конкурентоспроможність ігрового програмного застосунку на ринку ігор. Крім того, це сприятиме покращенню ігрового досвіду для гравців.

1.3 Постановка задачі

Для задоволення потреби гравців, які бажають провести час у веселій та соціально активній грі, а також отримати геймплей, спрямований на соціальну

взаємодію, необхідно розробити ігровий програмний застосунок у жанрі multiplayer party games. Це включатиме створення back-end частини системи та налаштування мережевої гри.

Ігровий програмний застосунок матиме назву ROFLSFUN, а ігровий процес буде організовуватися під час зборів друзів, де гравці використовуватимуть цифрові пристрої, такі як смартфони, планшети або інші з доступом до Інтернету та браузера для керування ігровим процесом. Результати їх керування та сам ігровий процес будуть транслюватися на великому екрані або моніторі, підключеному до комп'ютера чи іншого пристрою з Інтернетом і браузером. Ігровий програмний застсоунок буде складатися з трьох ігор: Brain Knights, Turing Test та Skibidy Party, – кожна з них є багатокористувацькою грою для вечірок та містить по одному з наступних жанрів: соціальна дедуктивна гра, гумористична карткова гра та змагальна вікторина. Вони будуть спрямовані на створення смішних та веселих моментів для учасників. Також ігровий програмний застосунок буде локалізовано на українську та англійську мови.

Skibidy Party – це багатокористувацька гра для вечірок у жанрі гумористичної карткової гри, в якій можуть брати участь від чотирьох до восьми гравців. Гра складається з кількох раундів, кількість яких дорівнює кількості учасників. Кожен раунд серед усіх гравців, які ще не були суддею раніше, випадковим чином вибирається суддя, який оцінює, на його думку, найкращі меми, обрані гравцями, відповідно до випадково обраної ситуації. Звичайні гравці, зі свого боку, отримують по шість неповторюваних карток з мемами, які не повторюються серед карток гравців поточного раунду або карток гравців попередніх раундів. Після того, як випадковим чином вибирається ситуація, гравці обирають найкумедніший мем зі своїх карток, який, на їхню думку, найкраще відображає обрану ситуацію. Після цього суддя отримує список мемів, вибраних гравцями, і розташовує їх у порядку від першого до третього місця. Гравець, який займає перше місце, отримує 1500 балів, друге місце – 1000 балів, третє – 500 балів. Перемагає той, хто набирає найбільшу кількість балів протягом усіх раундів.

Turing Test – це командна багатокористувацька гра для вечірок у жанрі соціально дедуктивної гри, в якій можуть брати участь від чотирьох до восьми гравців. Гра складається з трьох раундів. Гра розподіляє 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. Перед початком гри, коли гравці ще знаходяться у кімнаті, де ігрова сесія ще не почалася, вони можуть змінювати свою команду відповідно до співвідношення, якщо є вільні місця. Команда студентів починає гру, кожен її учасник задає запитання, на яке повинні відповісти професори та штучний інтелект. Коли професори надають свої відповіді, штучний інтелект отримує дані про запитання та варіанти відповідей професорів і намагається замаскувати свою відповідь під відповідь професорів. Після цього настає етап голосування, під час якого студенти визначають, яка з відповідей призначена штучному інтелекту. Якщо студенти успішно впізнають штучний інтелект, їхній загальний рахунок збільшується на два бали, в протилежному випадку один бал отримує команда професорів. Перемагає та команда, яка набрала більше балів. У випадку рівного рахунку гра закінчується в нічию. Мета команди студентів полягає в тому, щоб виявити штучний інтелект серед професорів, тоді як команда професорів намагається заплутати студентів, відповідаючи як штучний інтелект.

Brain Knights – це командна багатокористувацька гра для вечірок у жанрі змагальної вікторини. В ній можуть приймати участь від чотирьох до восьми гравців. Гра складається з двох етапів. Перший етап передбачає проведення серії раундів. Кількість раундів визначається як подвоєна максимальна кількість гравців серед двох команд, щоб кожен учасник міг взяти участь у цьому етапі принаймні двічі. У кожному раунді випадковим чином обирається тема. Потім капітани кожної команди вибирають по одному гравцю, який буде відповідати на запитання з обраної теми. Кожне запитання має чотири варіанти відповідей, з яких лише один є правильним. Підрахунок правильних відповідей здійснюється системою, і в кінці

кожного раунду учасник, який назбирав найбільшу кількість правильних відповідей, додає один бал своїй команді. У випадку, коли кількість правильних відповідей учасників однакова, обидва учасники додають по одному очку своїм командам. Під кінець першого етапу визначається найцінніший гравець гри – той, хто надав найбільшу кількість правильних відповідей. Другий етап полягає у проведенні командної вікторини, в якій беруть участь всі члени команди. Цей етап складається з трьох раундів. Перед кожним раундом капітани вибирають чотири теми, які будуть виключені. Питання стають складнішими, а командам надається певний час для обговорення кожного запитання, на яке є лише одна правильна відповідь. Кожна правильна відповідь надає команді один бал. Після завершення другого етапу визначається команда-переможець, яка здобула найбільшу кількість балів. У випадку рівної кількості балів встановлюється нічия.

У межах кваліфікаційної роботи на back-end частині системи необхідно охопити розробку логіки лише двох із вищеописаних ігор – соціальної дедуктивної гри Turing Test (без функціоналу, пов’язаного з ШІ) та гумористичної карткової гри Skibidy Party.

Back-end частина системи повинна виконувати всі задачі, що необхідні для правильної роботи застосунку, а також обробляти всі дії гравців, які передбачені геймплеєм та передбачають синхронізацію з back-end частиною. Це включатиме ідентифікацію гравців, створення і управління гральними кімнатами, синхронізацію гри між усіма гравцями, обробку дій гравців, обробку та збереження даних сесії гри протягом часу їх існування. Ця частина програмного забезпечення має забезпечити ефективну та безперебійну роботу ігрового середовища.

Також необхідно виконати налаштування мережевої гри, що дозволить гравцям підключатися до грального сесії та взаємодіяти один з одним у реальному часі. Для цього необхідно на клієнтській стороні реалізувати інтеграцію Socket.IO для налаштування WebSocket-з’єднання з метою забезпечення зв’язку між клієнтами та сервером [1]. На back-end частині необхідно створити WebSocket-сервер та використовувати бібліотеку Socket.IO для його налаштування, щоб забезпечити стабільний та швидкий обмін даними з клієнтською частиною [2].

Це дозволить клієнтській частині використовувати весь функціонал мережевої гри, який надає back-end частина.

Клієнтська частина буде використовувати бібліотеку React і буде реалізована на мові програмування JavaScript [3]. Для забезпечення зв’язку з back-end частиною слід інтегрувати Socket.IO, щоб виконати налаштування WebSocket-з’єднання з back-end частиною.

Back-end частина має бути розроблена з використанням програмної платформи Node.js, фреймворку NestJS та мови програмування TypeScript [4]. Для обробки HTTP-запитів слід обрати вбудовану платформу Express, що є стандартом для NestJS. Для забезпечення можливості обміну даними в реальному часі між клієнтом та сервером необхідно використати бібліотеку Socket.IO разом з інструментами WebSocket, що вбудовані в NestJS. Для збереження файлів зображень слід використовувати локальне файлове сховище, яке має бути розташоване на тому самому пристрої, де і знаходиться back-end частина. Зберігати статичні текстові дані, необхідні для роботи ігор, такі як питання, відповіді на них та опис ситуацій, слід українською та англійською мовами.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

У межах кваліфікаційної роботи розробка ігрового програмного застосунку у жанрі multiplayer party games включатиме створення back-end частини системи та налаштування мережевої гри. Причому розробка back-end частини охоплюватиме створення логіки лише двох із трьох ігор, які будуть доступні через застосунок – соціальної дедуктивної гри Turing Test (без функціоналу, пов'язаного з ШІ) та гумористичної карткової гри Skibidy Party.

Налаштування мережевої гри включає створення WebSocket-серверу та його налаштування з використанням бібліотеки Socket.IO на back-end частині системи. Також вона передбачає інтеграцію Socket.IO для налаштування WebSocket-з’єднання на клієнтській стороні з метою успішної комунікації між клієнтами та сервером. Це забезпечить можливість використовувати клієнтській стороні весь функціонал мережевої гри, який надається back-end частиною.

Налаштування мережевої гри виконується розробником і не є частиною функціоналу, який забезпечується програмною системою. Отже, для цієї операції не потрібно описувати функціональні та нефункціональні вимоги. В такому випадку опишемо їх для результату налаштування мережевої гри. Налаштована мережева гра повинна вирішувати наступні завдання: забезпечення обміну інформацією між браузером та веб-сервером у режимі реального часу за допомогою протоколу WebSocket.

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

– обробка всіх дій гравців, які передбачені геймплеєм та передбачають синхронізацію з back-end частиною;

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

– ідентифікація гравців;

– створення і управління гральними кімнатами;

– синхронізація гри між усіма гравцями;

– обробку та збереження даних сесії гри протягом часу їх існування;

– завантаження і управління файлами зображень;

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

Налаштована мережева гра ігрового програмного застосунку у жанрі multiplayer party games має наступні функціональні вимоги:

– повинна забезпечувати WebSocket-з’єднання між клієнтською та серверною сторонами;

– повинна забезпечувати можливість розірвання WebSocket-з’єднання з клієнтської сторони;

– повинна забезпечувати обмін даними між клієнтською та серверною сторонами через WebSocket-з’єднання.

Налаштована мережева гра ігрового програмного застосунку у жанрі multiplayer party games має наступні нефункціональні вимоги: повинна забезпечувати стабільне та безперервне функціонування WebSocket-з’єднання, яке має перериватися лише у випадку очікуваних подій, таких як, наприклад, втрата зв’язку з Інтернетом.

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

а) для 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, back-end 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;

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

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

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

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

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

11) повинна нараховувати бали команді переможців;

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

в) для всіх ігор разом:

1) повинна надавати можливість організатору створювати та входити в кімнату;

2) повинна надавати можливість гравцю входити в кімнату;

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

4) повинна надавати можливість гравцю виходити з кімнати;

5) повинна автоматично передати статус власника кімнати наступному в черзі гравцю, який увійшов після попереднього власника кімнати, котрий вийшов з кімнати;

6) повинна надавати можливість організатору змінювати мову в кімнаті;

7) повинна надавати можливість організатору встановлювати мову кімнати загальною для всіх її учасників;

8) повинна надавати можливість власнику кімнати розпочинати ігрову сесію для учасників кімнати;

9) повинна надавати можливість організатору ставити ігрову сесію на паузу та відновлювати її роботу;

10) повинна надавати можливість організатору виходити з кімнати;

11) повинна завершити ігрову сесію та очистити її дані, якщо організатор вийшов з кімнати;

12) повинна надавати можливість користувачу отримувати дані про кімнату, ігрову сесію в ній та інформацію про себе;

13) повинна розподіляти досягнення серед гравців, які вони здобули протягом гри;

14) повинна виконувати валідацію отриманих даних;

15) 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) повинна надавати можливість користувачу отримувати інформацію про час відображення поточного екрану.

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

– повинна підтримувати українську та англійську мови контенту, який є статичним для гри;

– повинна виконувати логування своїх помилок та дій гравців;

– повинна зберігати файли зображень в локальному файловому сховищі на тому самому пристрої, де знаходиться back-end частина.

Для створення з’єднання з сервером та передачі даних в реальному часі слід використовувати бібліотеку socket.io-client. Цю бібліотеку слід використовувати на клієнтській частині, яка буде реалізована з використанням бібліотеки React та мови програмування JavaScript.

Back-end частина має базуватися на програмній платформі Node.js та бути створена з використанням фреймворку NestJS та мови програмування TypeScript. Для обробки HTTP-запитів слід використати вбудовану платформу Express, яка є стандартом для NestJS. Для забезпечення можливості обміну даними в реальному часі між клієнтом та сервером необхідно використати бібліотеку Socket.IO разом з інструментами WebSocket, що вбудовані в NestJS.

Наприкінці написання цього пункту була створена специфікація для ігрового програмного застосунку (додаток В), яка містить детальний опис функціональних і нефункціональних вимог до програмного продукту. Завдяки специфікації забезпечується чітке розуміння очікуваних функцій, характеристик та обмежень системи, що мінімізує ризики непорозумінь та помилок під час розробки. Крім того, був складений календарний план до кваліфікаційної роботи, в якому послідовно описані етапи роботи над проєктом.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

Для ігрового програмного застосунку у жанрі multiplayer party games було створено діаграму розгортання (рис. 3.1). Вона описує аспект самої системи: фізичне розміщення інформації, що генерується програмою, на апаратних компонентах [5].

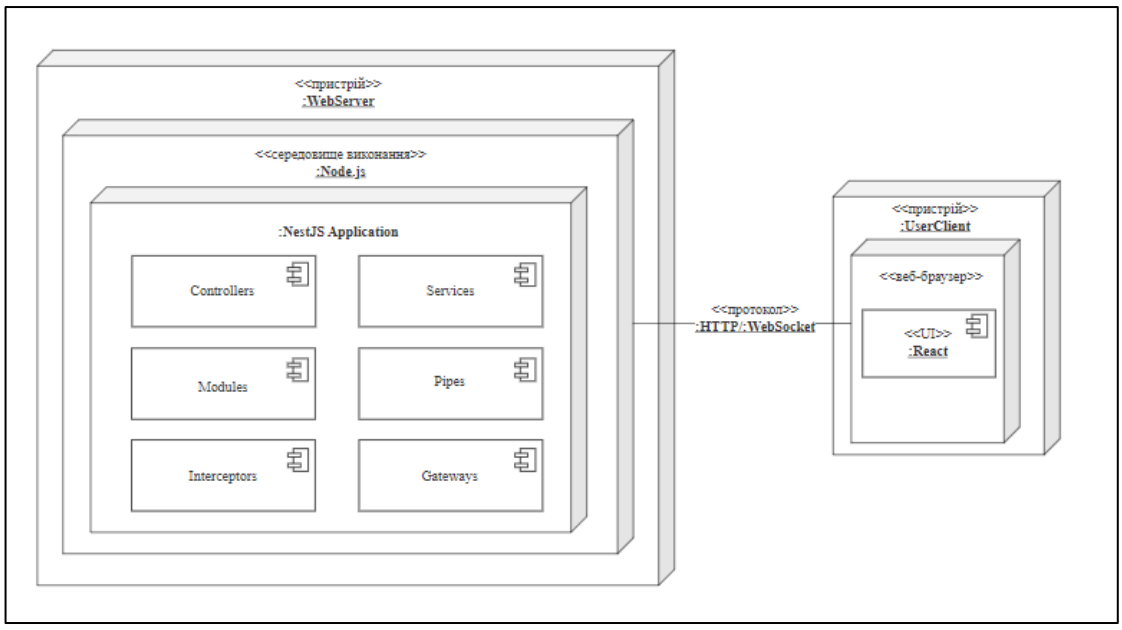


Рисунок 3.1 – UML діаграма розгортання ігрового програмного застосунку (діаграма виконана самостійно)

На діаграмі зображено архітектурну структуру програмного забезпечення та його фізичне розміщення на апаратних компонентах. Узлом у системі є веб-сервер, що використовує середовище виконання Node.js [6]. Усередині цього середовища знаходиться застосунок NestJS, який складається з різних компонентів, що є характерними для цього фреймворку [7].

Крім того, наведено клієнтську частину системи, яка представлена вузлом користувацького клієнта у веб-браузері. У веб-браузері ми маємо компонент інтерфейсу користувача, розроблений за допомогою React [8].

Ці узли пов’язані між собою через середовище виконання Node.js та веб-браузер за допомогою протоколу HTTP та WebSocket для обміну даними.

Важливо зазначити, що Node.js надає можливість об’єднати функції веб-сервера та сервера, який виконує бізнес-логіку та обробку даних, в одному середовищі виконання. Це означає, що не потрібно встановлювати окремий веб-сервер, такий як Apache або Nginx, а потім підключати до нього сервер додатків. Замість цього, можна створити веб-сервер безпосередньо з допомогою Node.js, що спрощує конфігурування та управління інфраструктурою.

Наступною є діаграма прецедентів, яка описує функціонал, доступний через клієнтську частину застосунку користувачу, що ще не ініціював ігрову сесію або не приєднався до існуючої (рис. 3.2).

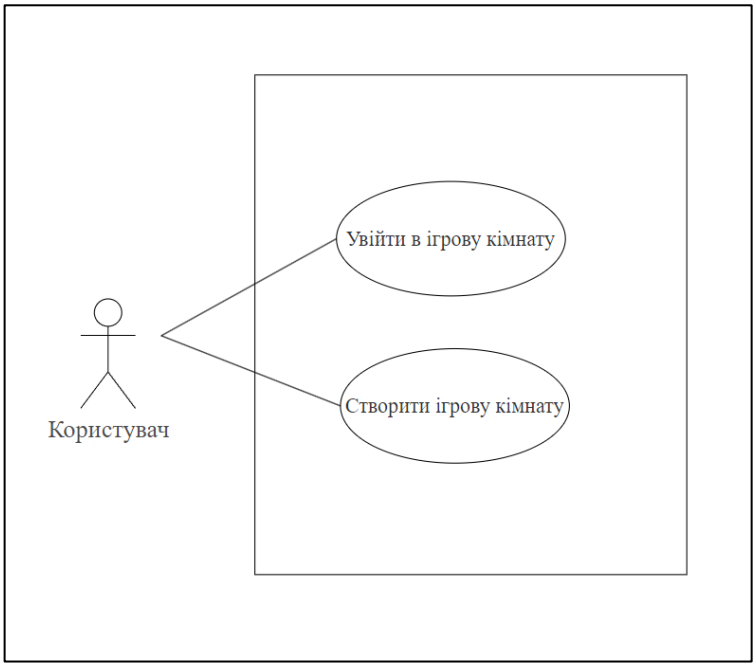


Рисунок 3.2 – UML діаграма прецедентів для користувача, який ще не приєднався до ігрової сесії (діаграма виконана самостійно)

Згідно цієї діаграми, він може взаємодіяти з системою наступним чином:

– увійти в ігрову кімнату;
– створити ігрову кімнату.

Наступна діаграма прецедентів описує функціонал, доступний через клієнтську частину застосунку для гравця з та без ролі власника кімнати, які знаходяться в кімнаті та очікують інших гравців (рис. 3.3).

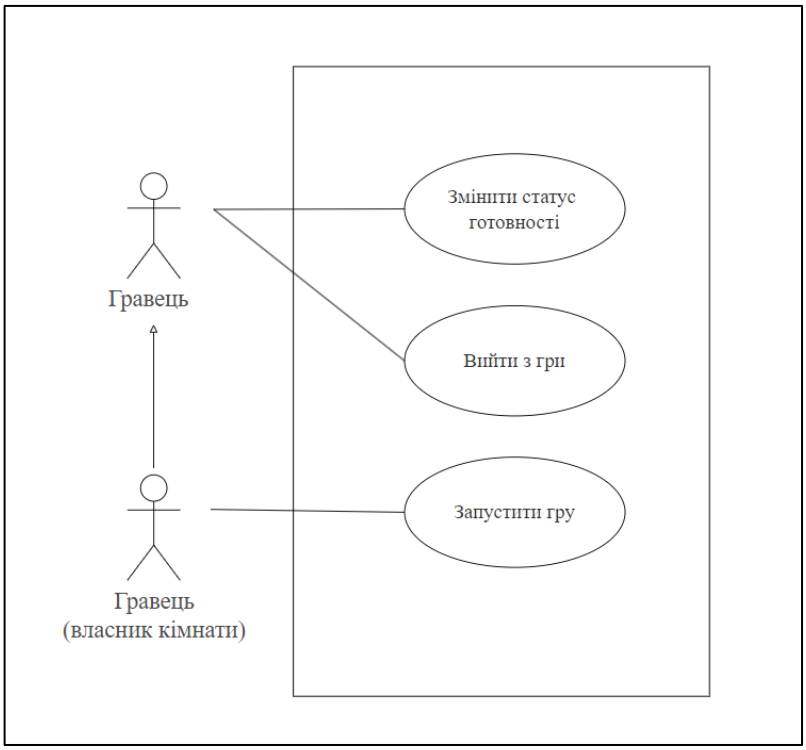


Рисунок 3.3 – UML діаграма прецедентів для гравців з та без ролі власника кімнати, які очікують інших гравців (діаграма виконана самостійно)

Згідно цієї діаграми, гравець без ролі власника кімнати може взаємодіяти з системою наступним чином:

– змінити статус готовності;

– вийти з гри.

Власник кімнати може взаємодіяти з системою так само, як і звичайний гравець, але крім того може запустити гру.

Наступна діаграма прецедентів описує функціонал, доступний через клієнтську частину застосунку для організаторів всіх ігор, а також розширений функціонал, доступний для організатора гри Skibidy Party (рис. 3.4).

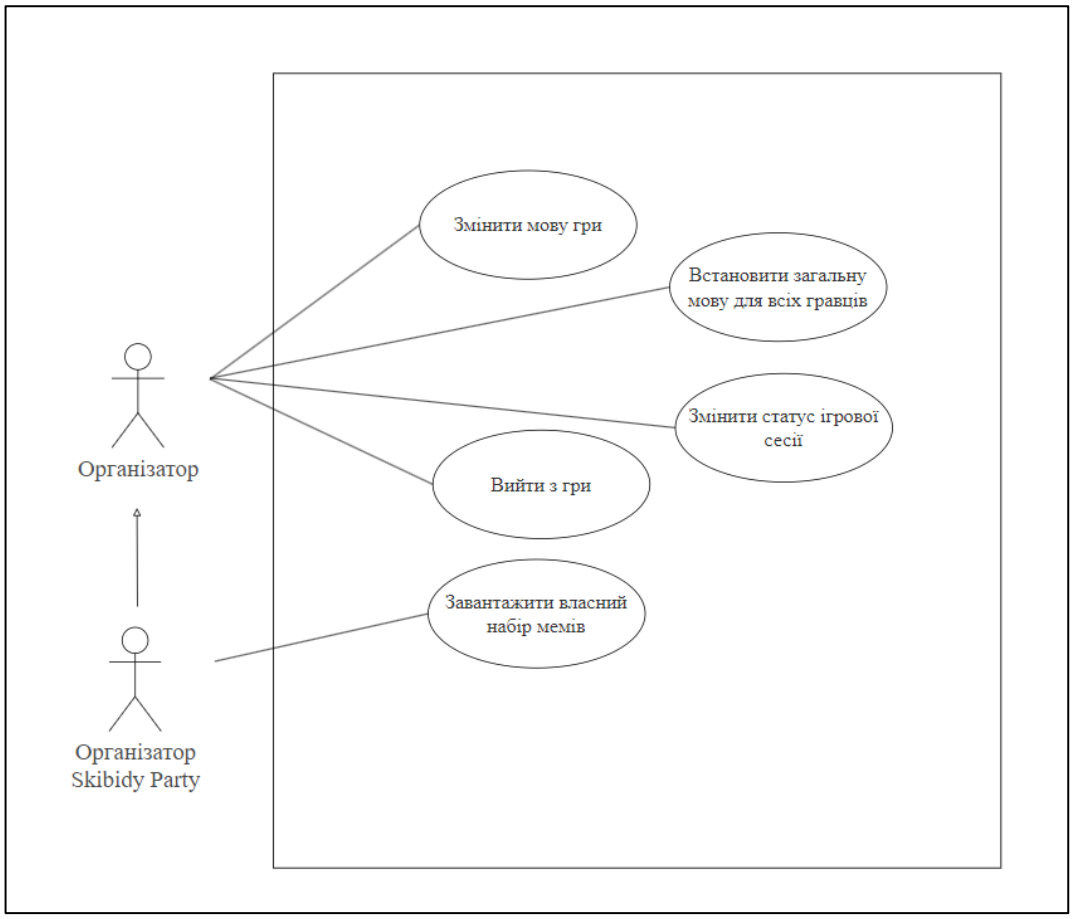


Рисунок 3.4 – UML діаграма прецедентів для організаторів всіх ігор, а також, для організатора гри Skibidy Party (діаграма виконана самостійно)

Згідно цієї діаграми, організатори всіх ігор можуть взаємодіяти з системою наступним чином:

– змінити мову гри;

– встановити загальну мову для всіх гравців;

– вийти з гри.

Організатор Skibidy Party може також завантажувати власний набір мемів для гри.

Наступна діаграма прецедентів описує функціонал, доступний через клієнтську частину застосунку для учасників гри Skibidy Party (рис. 3.5).

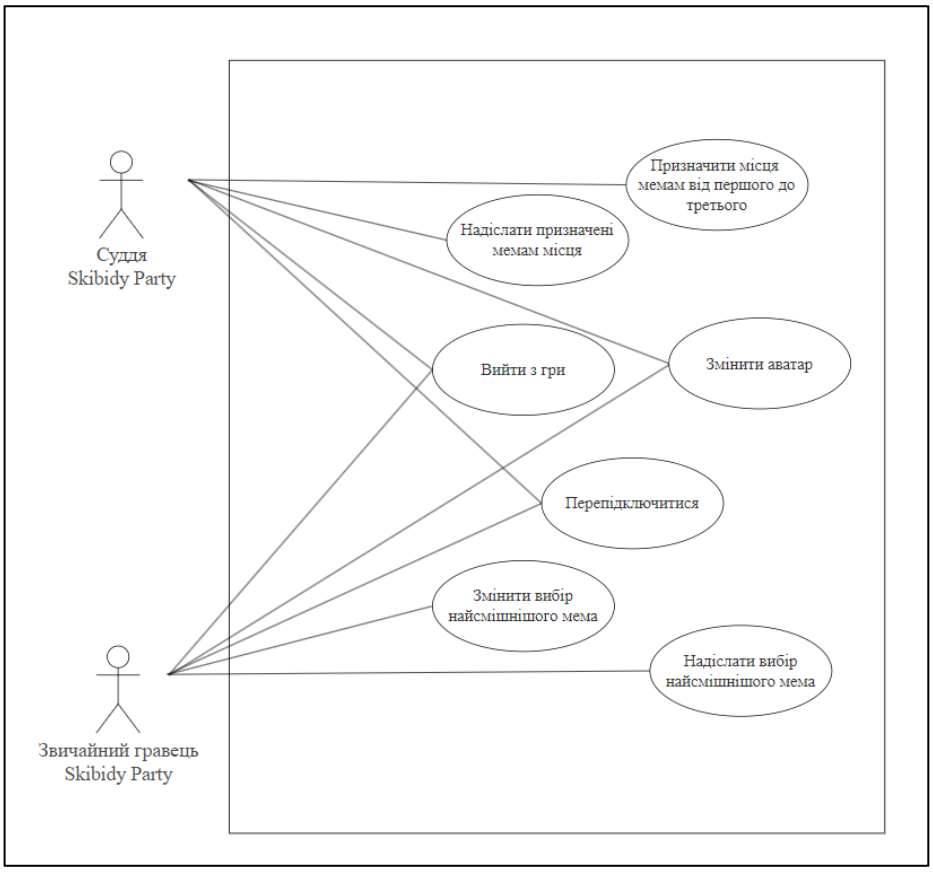


Рисунок 3.5 – UML діаграма прецедентів для гравців Skibidy Party (діаграма виконана самостійно)

Згідно цієї діаграми, суддя може взаємодіяти з системою наступним чином:

– призначити місця мемам від першого до третього;

– надіслати призначені мемам місця;

– змінити аватар;

– вийти з гри;

– перепідключитися.

Згідно цієї діаграми, звичайний гравець може взаємодіяти з системою наступним чином:

– змінити вибір найсмішнішого мема;

– надіслати вибір найсмішнішого мема;

– змінити аватар;

– вийти з гри;

– перепідключитися.

Наступна діаграма прецедентів описує функціонал, доступний через клієнтську частину застосунку для учасників гри Turing Test (рис. 3.6).

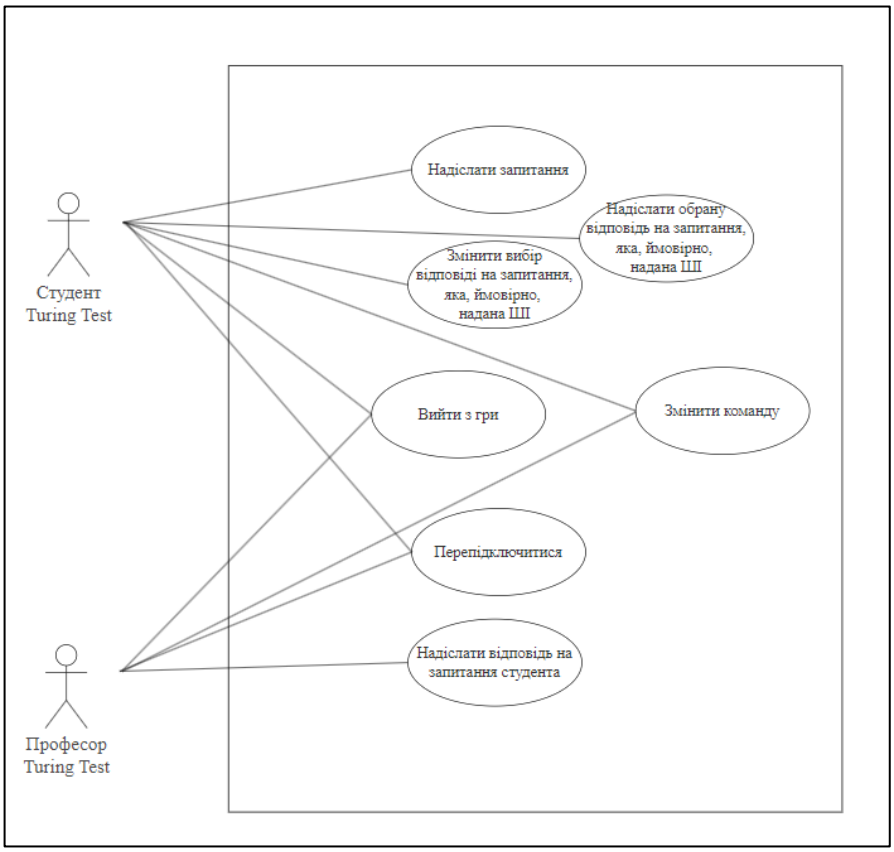


Рисунок 3.6 – UML діаграма прецедентів для гравців Turing Test (діаграма виконана самостійно)

Згідно цієї діаграми, студент може взаємодіяти з системою наступним чином:

– надіслати запитання;
– надіслати обрану відповідь на запитання, яка, ймовірно, надана ШІ;
– змінити вибір відповіді на запитання, яка, ймовірно, надана ШІ;
– змінити команду;
– вийти з гри;
– перепідключитися.

Згідно цієї діаграми, професор може взаємодіяти з системою наступним чином:

– надіслати відповідь на запитання студента;
– змінити команду;

– вийти з гри;

– перепідключитися.

Наступною є загальна діаграма пакетів для всього програмного застосунку (рис. 3.7). Вона відображає глобальні пакети програми, а також сторонній сервіс, який є необхідним для його функціонування.

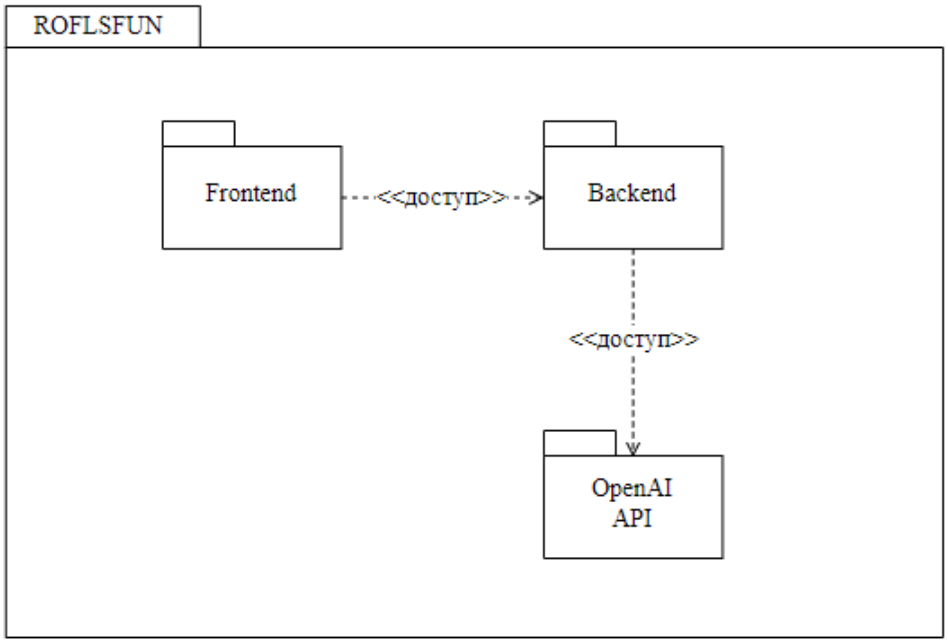


Рисунок 3.7 – Загальна UML діаграма пакетів ігрового програмного застосунку (діаграма виконана самостійно)

Наступна діаграма пакетів відображає взаємозв’язок між усіма NestJS модулями back-end частини ігрового програмного застосунку (рис. 3.8). Вона показує залежності кожного модулю один від одного. Модулі використовуються для організації коду, розділу функцій на логічні блоки, які можна повторно використовувати [9]. У цьому застосунку модулі ігор Turing Test, Brain Knights, Skibidy Party містять наступних постачальників: сервіси та шлюзи. Модуль Skibidy Party містить також контролер, який необхідний для завантаження зображень.

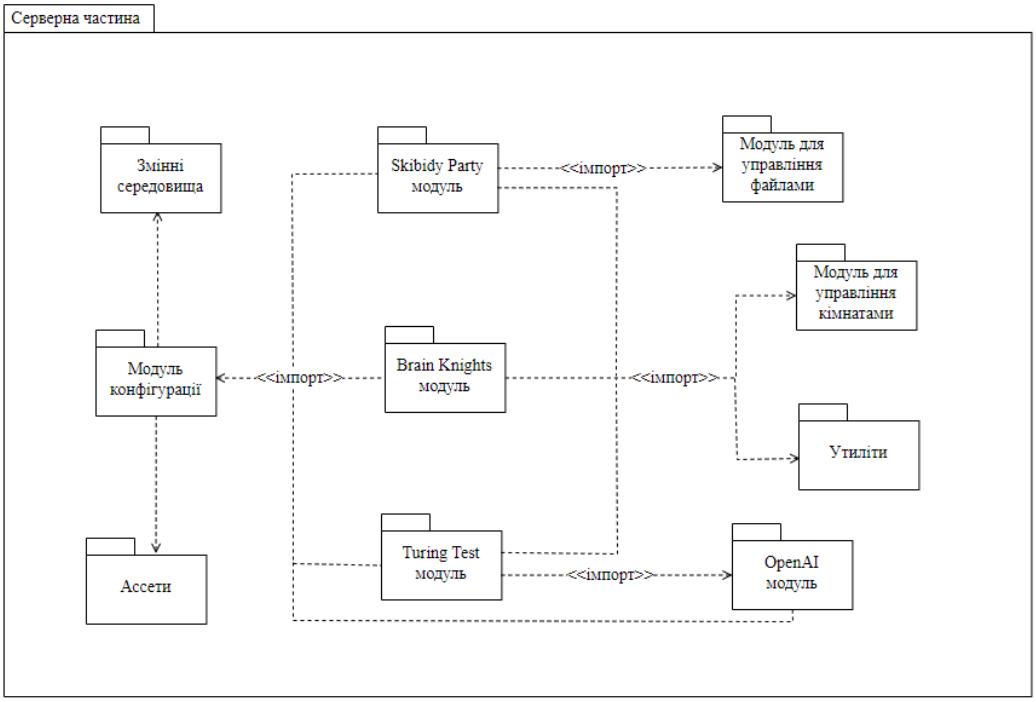


Рисунок 3.8 – UML діаграма пакетів ігрового програмного застосунку (діаграма виконана самостійно)

Наступною є діаграма діяльності, яка відображає послідовність подій на back-end частині після отримання повідомлення про ініціалізацію ігрової сесії (рис. 3.9).

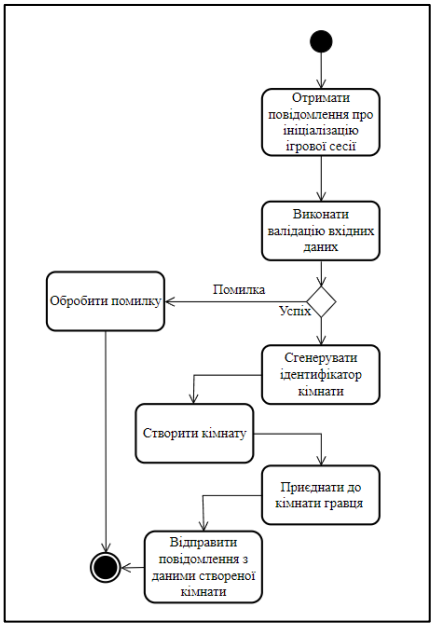


Рисунок 3.9 – UML діаграма діяльності отримання повідомлення про ініціалізацію ігрової сесії (діаграма виконана самостійно)

Наступна діаграма діяльності відображає послідовність подій на back-end частині після отримання повідомлення про вхід в кімнату користувача (рис. 3.10).

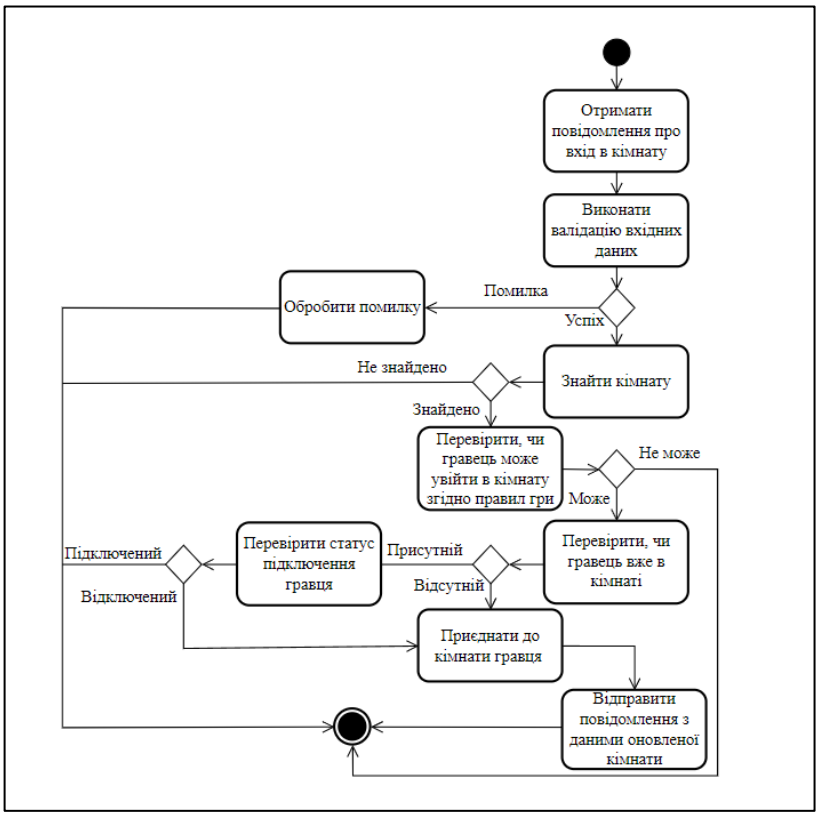


Рисунок 3.10 – UML діаграма діяльності отримання повідомлення про вхід в кімнату (діаграма виконана самостійно)

Для того, щоб почати гру, гравцю необхідно, щоб виконувалися наступні умови:

– дані, передані ним, були валідні;

– була наявна кімната;

– в гру можна увійти згідно з її правилами;

– гравець відсутній в кімнаті, або присутній, але відключений.

Наступна діаграма діяльності відображає послідовність подій на back-end частині після отримання повідомлення про інверсію статусу готовності гравця, який знаходиться в кімнаті та очікує приєднання інших гравців або їх готовності до гри (рис. 3.11).

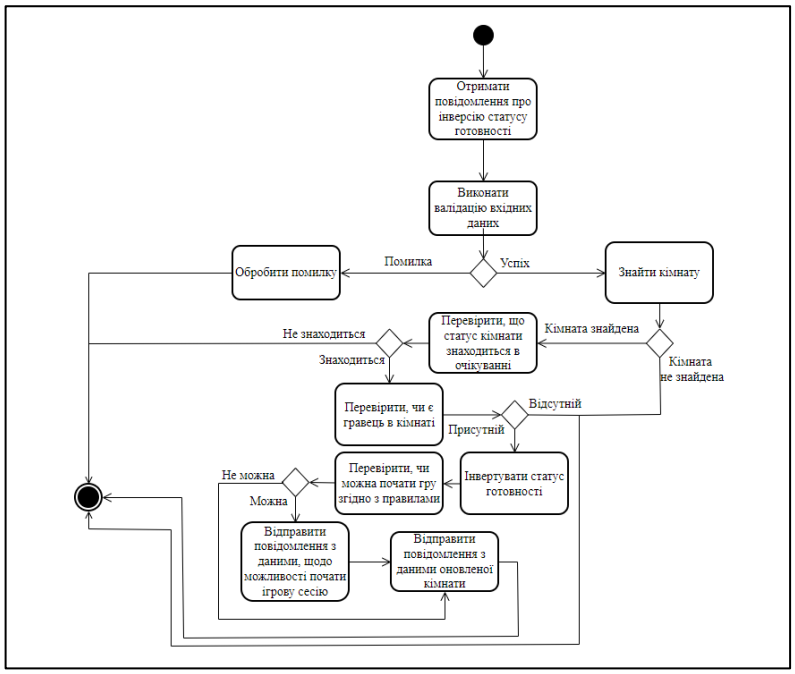


Рисунок 3.11 – UML діаграма діяльності отримання повідомлення про інверсію статусу готовності (діаграма виконана самостійно)

Наступна діаграма діяльності відображає послідовність подій на back-end частині після отримання повідомлення про вихід користувача з кімнати (рис. 3.12).

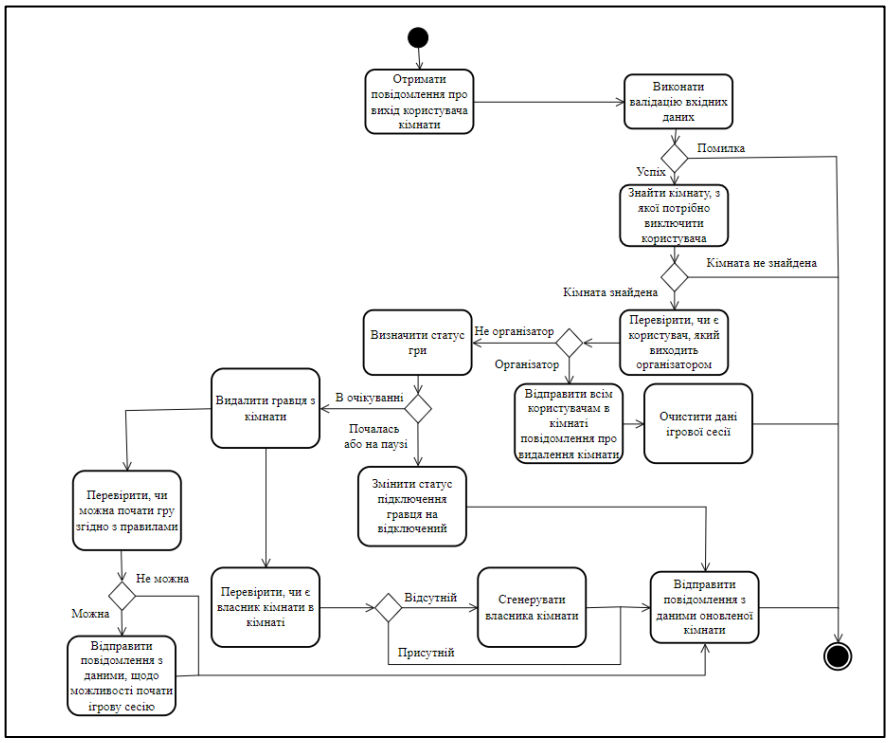


Рисунок 3.12 – UML діаграма діяльності отримання повідомлення про вихід користувача з кімнати (діаграма виконана самостійно)

Наступна діаграма діяльності відображає послідовність подій на back-end частині після отримання повідомлення про спробу почати гру (рис. 3.13).

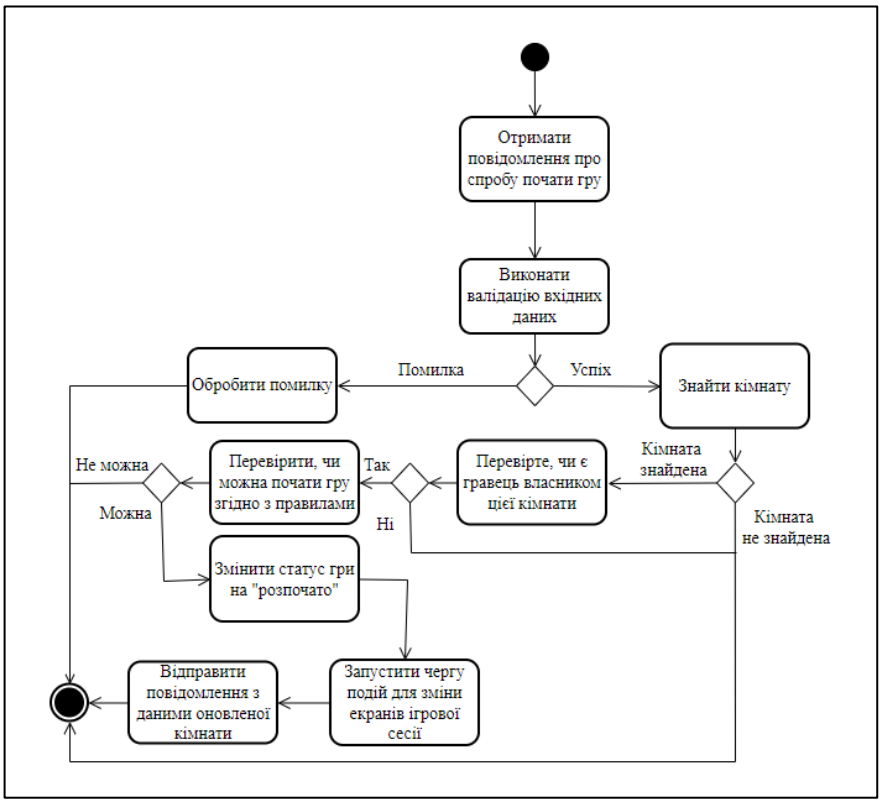


Рисунок 3.13 – UML діаграма діяльності отримання повідомлення про спробу почати гру (діаграма виконана самостійно)

Для того, щоб почати гру, гравцю необхідно, щоб виконувалися наступні умови:

– дані, передані ним, були валідні;

– була наявна кімната;

– він був власником цієї кімнати;

– гру можна було почати згідно з її правилами.

Наступна діаграма діяльності відображає послідовність подій на back-end частині після отримання повідомлення про зміну мови кімнати організатором (рис. 3.14).

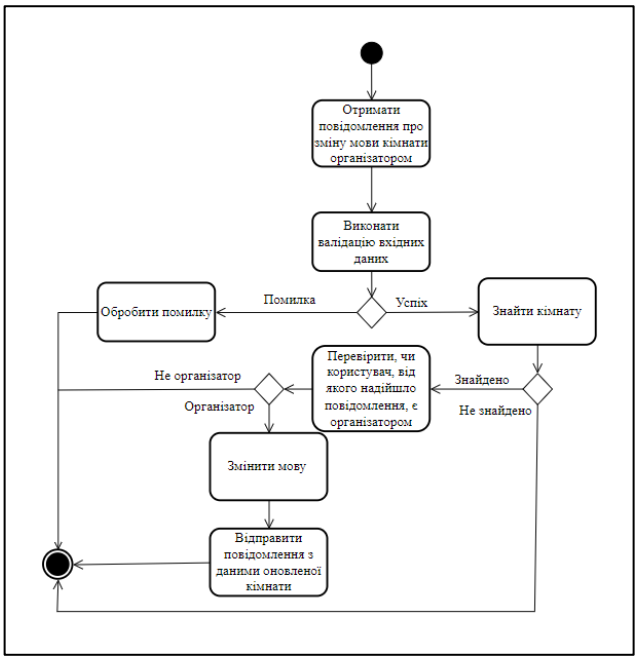


Рисунок 3.14 – UML діаграма діяльності отримання повідомлення про зміну мови кімнати організатором (діаграма виконана самостійно)

Наступна діаграма діяльності відображає послідовність подій на back-end частині після отримання повідомлення про встановлення загальної мови кімнати (рис. 3.15).

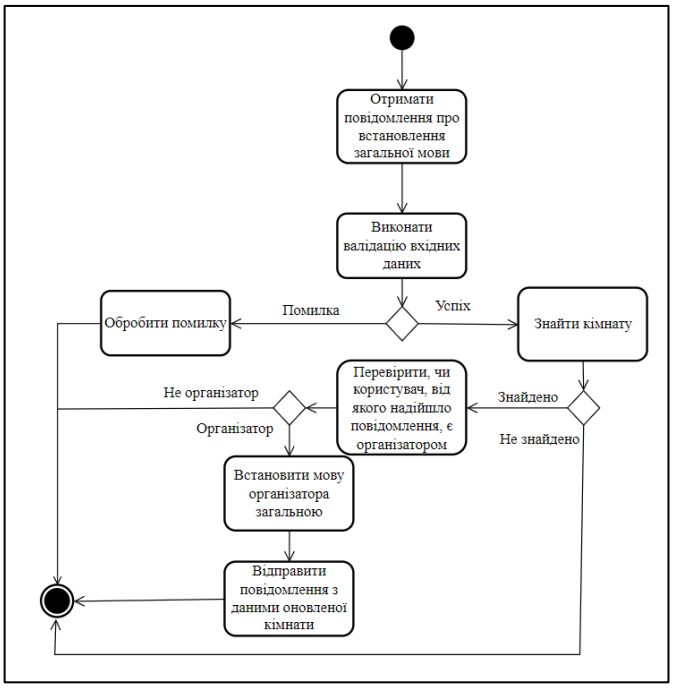


Рисунок 3.15 – UML діаграма діяльності отримання повідомлення про встановлення загальної мови кімнати (діаграма виконана самостійно)

Наступна діаграма діяльності відображає послідовність подій на back-end частині після отримання повідомлення про зміну статусу ігрової сесії (рис. 3.16).

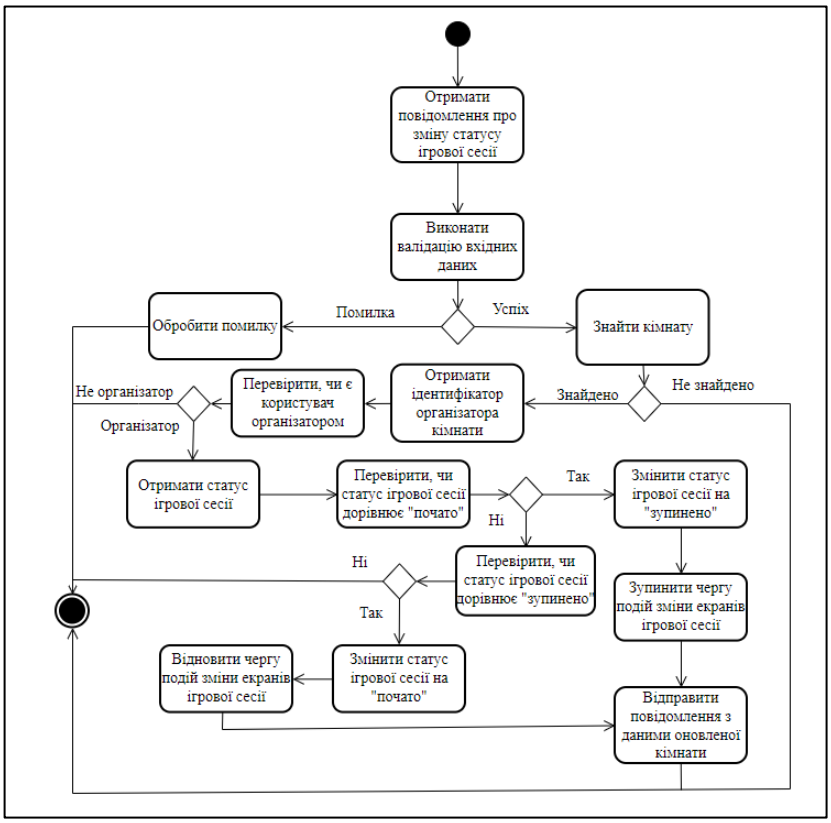


Рисунок 3.16 – UML діаграма діяльності отримання повідомлення про зміну статусу ігрової сесії (діаграма виконана самостійно)

Наступна діаграма діяльності відображає послідовність подій на back-end частині після отримання повідомлення про призначення мему місця (рис. 3.17). Для зручності перегляду діаграми вона поділена на 5 частин (рис. 3.18-3.22).

Частини під номерами 1 та 2 описують послідовність перевірки можливості присвоєння будь-якого місця мему. Частини з 3 по 5 описують процес надання конкретного місця обраному суддею мему.

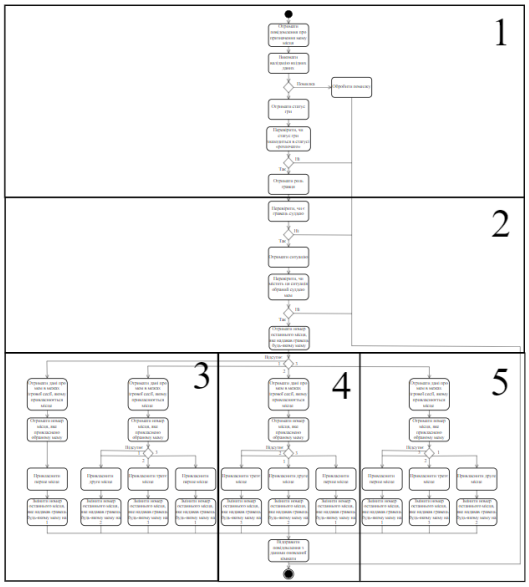


Рисунок 3.17 – UML-діаграма діяльності отримання повідомлення про призначення мему місця в грі Skibidy Party (діаграма виконана самостійно)

Перша частина UML-діаграми діяльності отримання повідомлення про призначення мему місця в грі Skibidy Party відображає первинну послідовність подій, де виконується валідація даних та перевіряється статус ігрової сесії.

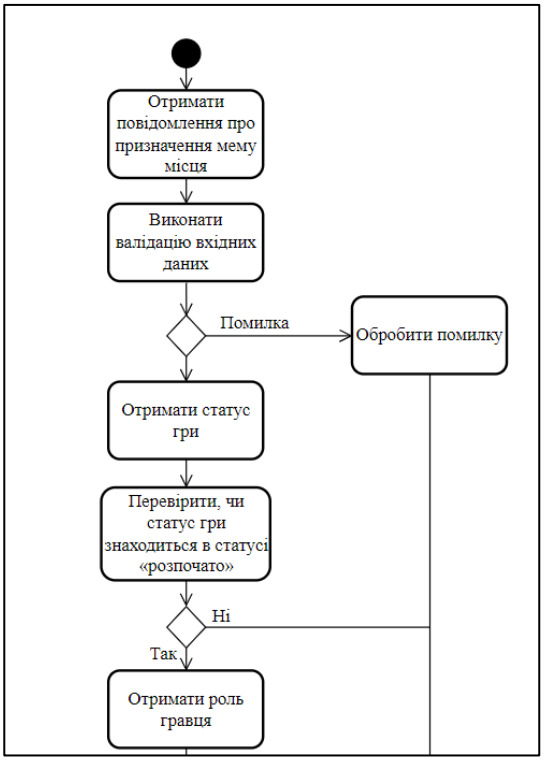


Рисунок 3.18 – Частина 1 UML-діаграми діяльності отримання повідомлення про призначення мему місця в грі Skibidy Party (діаграма виконана самостійно)

Друга частина діаграми також відображає первинну послідовність подій, де виконується перевірка, чи є гравець суддею, чи призначав він вже цьому мему місце, і якщо так, то відображається подія отримання цього місця.

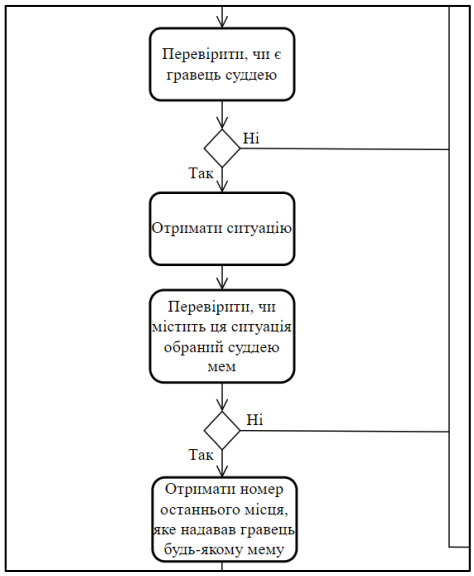


Рисунок 3.19 – Частина 2 UML-діаграми діяльності отримання повідомлення про призначення мему місця в грі Skibidy Party (діаграма виконана самостійно)

Третя частина діаграми відображає послідовність подій, якщо мему не було призначено місце або було призначено перше місце

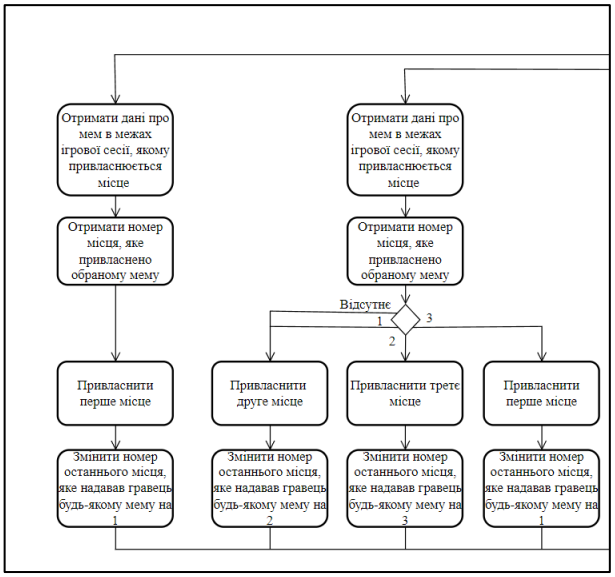


Рисунок 3.20 – Частина 3 UML-діаграми діяльності отримання повідомлення про призначення мему місця в грі Skibidy Party (діаграма виконана самостійно)

Четверта частина діаграми відображає послідовність подій, якщо мему було призначено друге місце.

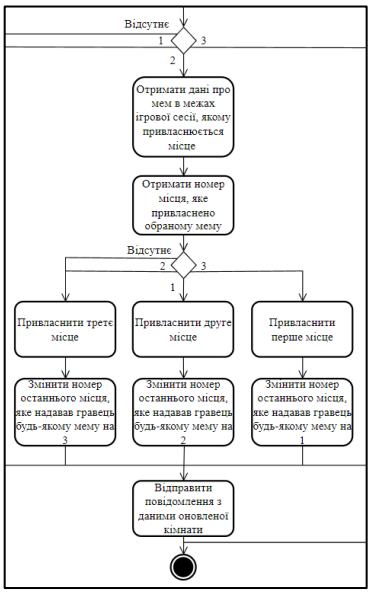


Рисунок 3.21 – Частина 4 UML-діаграми діяльності отримання повідомлення про призначення мему місця в грі Skibidy Party (діаграма виконана самостійно)

П’ята частина діаграми відображає послідовність подій, якщо мему було призначено третє місце.

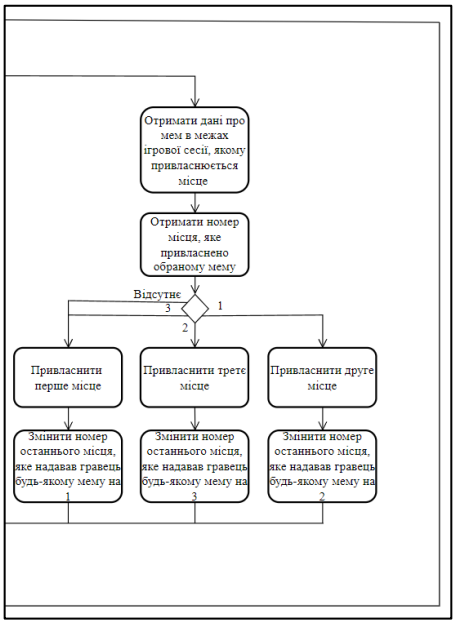


Рисунок 3.22 – Частина 5 UML-діаграми діяльності отримання повідомлення про призначення мему місця в грі Skibidy Party (діаграма виконана самостійно)

Використання діаграм UML сприяє кращому розумінню взаємодії між системою та її оточенням, вимог до функцій та функціональності системи. Ці діаграми дозволяють візуалізувати зв’язки між різними компонентами системи та їх взаємодію. Крім того, за використання UML дозволяє визначати структуру програмного забезпечення та взаємодію між його компонентами на високому рівні. Це сприяє кращому розумінню того, як система має функціонувати перед тим, як розпочати фактичний процес розробки.

3.2 Проєктування архітектури ПЗ

У якості архітектури, на якій базується back-end частина застосунку, була обрана трьохрівнева архітектура (рис. 3.23). Перевага полягає в тому, що вона дозволяє легко змінювати кожен рівень незалежно один від одного, а також робить систему гнучкішою та легшою в підтримці.

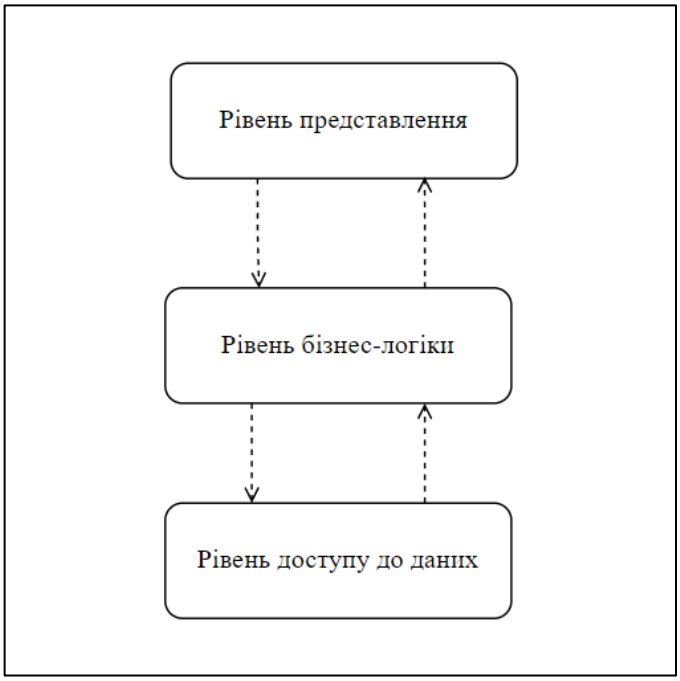


Рисунок 3.23 – Архітектура back-end частини застосунку (діаграма виконана самостійно)

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 будуть розміщені контролери та шлюзи, для створення яких використовується фреймворк NestJS. Цей рівень не повинен містити логіки бізнес-процесів або доступу до даних.

Контролери у NestJS використовуються для обробки HTTP-запитів та відповідей. Основна їх функція полягає в прийомі вхідних HTTP-запитів від клієнтів, обробці їх та поверненні відповідей. Шлюзи використовуються для обробки та маршрутизації WebSocket-запитів.

Рівень бізнес-логіки відповідає за обробку даних і бізнес-процесів програми. Тут будуть розміщені сервіси, які будуть виконувати операції, пов’язані з обробкою та зміною даних відповідно до бізнес-правил. Цей рівень не буде мати прямого доступу до даних, але він зможе отримувати дані з шару представлення і передавати їх на рівень доступу до даних для збереження або вилучення.

Back-end частина ігрового програмного застосунку у жанрі multiplayer party games також буде мати рівень доступу до даних. Однак, оскільки дані кімнат зберігаються в пам’яті застосунку, цей рівень буде представляти собою набір сервісів та класів, які забезпечуватимуть доступ до них. Цей рівень не буде містити бізнес-логіки, він буде складатися лише з операцій, які безпосередньо пов’язані з доступом до даних.

Структуру back-end частини можна представити як структуру застосунку, побудованого на базі фреймворку NestJS (рис. 3.24).

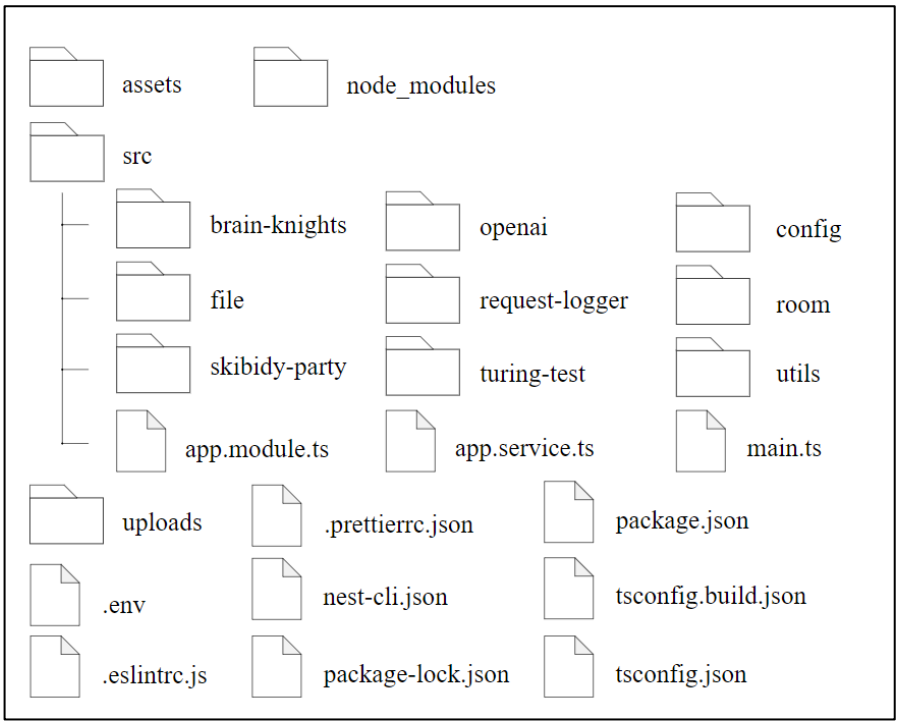


Рисунок 3.24 – Структура back-end частини (рисунок виконано самостійно)

Окрім стандартних файлів, що характерні для фреймворку NestJS та програмної платформи Node.js, на рисунку зображено наступні каталоги:

– assets: цей каталог заплановано для зберігання файлів із статичним контентом гри, таких як запитання, відповіді та смішні ситуації;

– brain-knights: він міститиме код, що відповідає за функціонування гри Brain Knights, а також необхідний код для виклику цієї логіки з клієнтської частини;

– openai: в цьому каталозі буде код, який забезпечує взаємодію з API OpenAI;

– config: тут будуть зберігатися файли, пов’язані з конфігурацією застосунку;

– file: в цьому каталозі буде розташований код, що відповідає за роботу з файлами;

– request-logger: цей каталог буде містити код для логування запитів, що надійшли від клієнта;

– room: тут буде знаходитися код, який відповідає за операції, що безпосередньо пов’язані з доступом до даних кімнат;

– skibidy-party: він буде містити код для функціонування гри Skibidy Party та код, необхідний для взаємодії з клієнтською частиною;

– turing-test: тут буде зберігатися код, який відповідає за функціонування гри Turing Test та необхідний код для взаємодії з клієнтською частиною;

– utils: цей каталог буде містити допоміжні інструменти.

Така структура повинна дозволити розмістити файли з кодом, який логічно пов’язаний в окремі папки, що сприятиме структуруванню проєкту та полегшить подальшу підтримку.

3.3 Приклади найцікавіших алгоритмів та методів

Перед початком будь-якої гри в ігровому програмному застосунку необхідно сформувати 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. Необхідно мати можливість змінювати чергу екранів та її елементи, а також ті елементи, які вже вийшли з неї, при необхідності, що випливають з логіки гри;

Для реалізації такого програмного коду можна створити клас, який включатиме наступні поля: черга, яка базується на імплементація масиву; множина елементів поточного елементу черги, який виконується і вже вийшов з черги; інтервали для кожного з множини елементів поточного елементу черги, який виконується і вже вийшов з черги; змінна, що відповідає за стан паузи [10].

Елементи черги повинні складатися з множини елементів, які включають наступні поля:

– ідентифікатор тайм-аута;

– функція зворотного виклику для тайм-ауту;

– функція зворотного виклику для інтервалу;

– час затримки тайм-аута;

– час, що залишився до виклику функції зворотного виклику тайм-ауту;

– час запуску тайм-ауту;

– ідентифікатор тайм-аута, який має виконатися перед виконанням основного тайм-аута;

– функція зворотного виклику для тайм-ауту, який має виконатися перед виконанням основного тайм-аута;

– час затримки тайм-аута, який має виконатися перед виконанням основного тайм-аута.

Після цього доповнити клас методами, які виконують наступний функціонал:

– встановлює новий інтервал за заданим ключем;

– очищає інтервал за заданим ключем;

– перевіряє, чи існує тайм-аут з вказаним ключем;

– очищає тайм-аут за заданим ключем;

– запускає виконання черги відображення екранів гри;

– додає нові множини тайм-аутів до черги;

– додає нові множини тайм-аутів до початку черги;

– виконує тайм-аути останнього елементу з черги відображення екранів;

– встановлює тайм-аут без зворотних викликів;

– встановлює раптовий тайм-аут за заданими параметрами;

– повертає час, який залишився до закінчення тайм-ауту;

– встановлює паузу;

– відновлює виконання елементів з черги;

– оновлює затримку для всіх тайм-аутів;

– очищає всі тайм-аути, черги та інтервали;

– перевіряє, чи можна встановити раптовий тайм-аут;

– очищає всі інтервали.

Серед цих методів можна виділити наступні: метод, який встановлює паузу, та метод, який відновлює виконання елементів з черги.

Серед цих методів можна виділити наступні: метод, який встановлює паузу, та метод, який відновлює виконання елементів з черги. Алгоритм виконання методу, який встановлює паузу, можна описати наступним чином:

– для кожного тайм-ауту зі списку: скасувати виконання тайм-аута, щоб припинити його дію; обчислити, скільки часу пройшло з моменту його запуску до моменту паузи; оновити залишковий час тайм-ауту, віднімаючи пройдений час від загального часу тайм-ауту; якщо існує тайм-аут, який має виконатися перед виконанням основного тайм-ауту, то скасувати його виконання;

– продовжити виконання цих дій для кожного тайм-ауту в списку, доки всі тайм-аути не будуть поставлені на паузу;

– змінити поле класу, що відповідає за паузу, позначивши, що всі тайм-аути встановлені на паузу.

Алгоритм виконання методу, який відновлює виконання елементів з черги, можна описати наступним чином:

а) для кожного тайм-ауту зі списку:

1) перевірити, чи залишився ще час до закінчення тайм-аута;

2) якщо час залишився:

– встановити новий тайм-аут з часом, що дорівнює часу, який залишився до закінчення тайм-аута;

– оновити час початку таймера, щоб врахувати час після паузи;

– якщо існує тайм-аут, який має виконатися перед виконанням основного тайм-ауту, то виконати наступні дії: обчислити новий залишковий час для нього, віднімаючи час, що залишився, від часу до завершення; оновити його з новим часом;

б) змінити поле класу, що відповідає за паузу, позначивши, що всі тайм-аути були відновлені.

Ці два методи є дуже важливими, оскільки реалізують механіку паузи та відновлення в грі.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

Back-end частина ігрового програмного застосунку розроблена на платформі Node.js з використанням фреймворку NestJS і мови програмування TypeScript. Для обробки HTTP-запитів обрано вбудовану платформу Express, що є стандартом для NestJS. Для забезпечення можливості обміну даними в реальному часі між клієнтом та сервером використано бібліотеку Socket.IO разом з інструментами WebSocket, що вбудовані в NestJS. Список бібліотек та їх версій, які використовуються в back-end частині ігрового програмного застосунку, наведено в таблиці 4.1.

Таблиця 4.1 – Список бібліотек та їх версій, які використовуються в back-end частині ігрового програмного застосунку (таблиця виконана самостійно)

Назва	Версія	Призначення
@nestjs/common	10.0.0	Використовуються для створення веб-додатків заснованих на фреймворку Nest.js. Вони надають базові функціональності та інтеграцію з Express, Socket.io та іншими платформами
@nestjs/config	3.1.1	
@nestjs/core	10.0.0	
@nestjs/platform-express	10.0.0	
@nestjs/platform-socket.io	10.2.6	
@nestjs/serve-static	4.0.0	
@nestjs/websockets	10.2.6	
axios	1.6.2	Використовується для виконання HTTP-запитів з Node.js
class-transformer	0.5.1	Використовуються для валідації та перетворення об’єктів в TypeScript
class-validator	0.14.0	
cookie-parser	1.4.6	Проміжне програмне забезпечення для розбору cookie в HTTP-запитах
crypto	1.0.1	Використовується для генерації унікальних назв файлів

Кінець таблиці 4.1

Назва	Версія	Призначення
helmet	7.1.0	Проміжне програмне забезпечення для захисту веб-додатків від різних атак
joi	17.11.0	Бібліотека для валідації даних
multer	1.4.5	Проміжне програмне забезпечення для обробки файлових завантажень у Node.js
socket.io	4.7.5	Бібліотека для роботи з WebSocket у Node.js.
uuid	9.0.1	Модуль для генерації унікальних ідентифікаторів

Для правильного функціонування back-end частини та логіки ігор Skibdy Party та Turing Test (без функціоналу, пов'язаного з ШІ) використовуються наступні змінні середовища:

– NODE_ENV: визначає середовище, в якому працює застосунку;

– PORT: вказує порт, на якому буде запущений сервер застосунку.;

– PROTOCOL: визначає протокол, який буде використовуватися для з’єднань;

– DOMAIN: вказує домен, на якому працює клієнтська частина застосунку;

– SKIBIDY_PARTY_PROPERTIES_PATH: шлях до файлу з властивостями для гри Skibidy Party; цей файл містить конфігураційні налаштування для цієї гри;

– TURING_TEST_PROPERTIES_PATH: шлях до файлу з властивостями для гри Turing Test; цей файл містить конфігураційні налаштування для цієї гри;

– STATIC_ROOT_FOLDER: вказує на папку, де зберігаються статичні файли застосунку;

– UPLOADS_FOLDER: визначає папку, в яку будуть зберігатися завантажені файли.

Файл з властивостями гри Skibidy Party (рис. 4.1) містить такі записи: ідентифікатор гри, назва гри, мінімальна необхідна кількість гравців, максимальна кількість гравців, аватари гравців, список ситуацій та мемів.

Рисунок 4.1 – Вміст файлу з властивостями гри Skibidy Party (знімок екрана виконано самостійно)

Ситуації містять масив об’єктів (рис. 4.2). Кожен об'єкт містить ідентифікатор та опис ситуації англійською та українською мовами. Для генерації набору смішних ситуацій двома мовами використовувався ChatGPT [11].

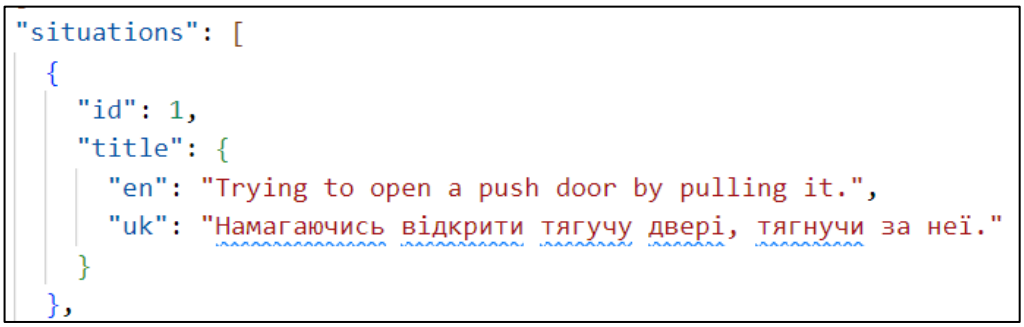


Рисунок 4.2 – Опис об'єкта ситуації у файлі з властивостями гри Skibidy Party (знімок екрана виконано самостійно)

Меми містять об’єкт (рис. 4.2), який включає шлях до них, шлях до стандартних мемів, максимальну кількість завантажуваних мемів, назву поля завантажуваних файлів, стандартний тип файлів та масив об'єктів, які описують стандартні файли мемів, вже завантажені на сервер.

Рисунок 4.3 – Опис об'єкта мема у файлі з властивостями гри Skibidy Party (знімок екрана виконано самостійно)

Масив, що описує файли, містить об’єкти (рис. 4.4), які містять ідентифікатор, підпис та назву файлу

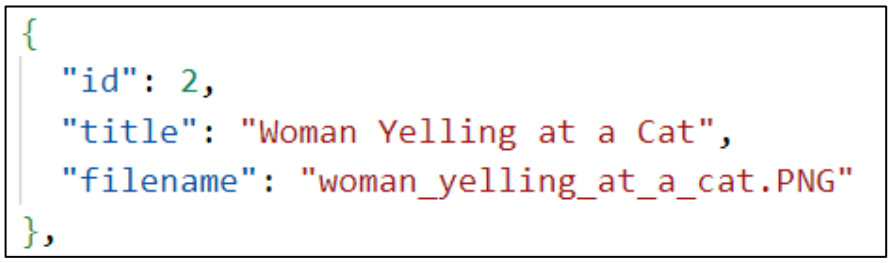


Рисунок 4.4 – Опис об'єкта файла у файлі з властивостями гри Skibidy Party (знімок екрана виконано самостійно)

Файл з властивостями гри Turing Test (рис. 4.1) містить такі записи: ідентифікатор гри, назва гри, мінімальна необхідна кількість гравців, максимальна кількість гравців.

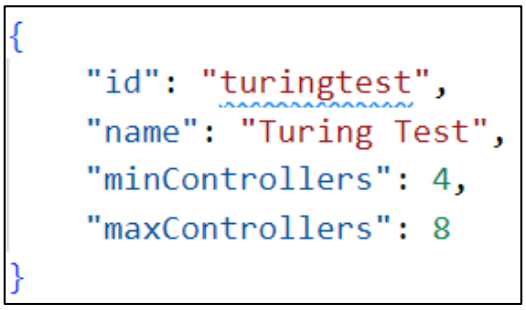


Рисунок 4.5 – Вміст файлу з властивостями гри Turing Test (знімок екрана виконано самостійно)

Оскільки всі ці конфігураційні значення змінюються залежно від середовища виконання застосунку (локальне середовище та виробниче середовище), то змінні конфігурації будуть зберігатися в середовищі. Для цього, використовуючи фреймворк NestJS, а саме бібліотеку @nestjs/config, було імпортовано ConfigModule, який налаштовано на використання власного файлу конфігурації:

```
@Module({
  imports: [
    ConfigModule.forRoot({
      isGlobal: true,
      validationSchema,
      load: [config],
    }),
    ServeStaticModule.forRoot({
      rootPath: join(__dirname, '..', process.env.STATIC_ROOT_FOLDER),
    }),
    RoomModule,
    BrainknightsModule,
    TuringTestModule,
    FileModule,
    SkibidyPartyModule,
  ],
  providers: [AppService],
})
```

Окрім того, в цей модуль імпортується модуль для обслуговування статичних файлів, а також модулі, які необхідні для роботи логіки ігор.

Власний файл конфігурації повертає об’єкт, який містить об'єднані оброблені дані змінних середовища та змінних властивостей ігор:

```
return {
  static: {
      rootFolder: process.env.STATIC_ROOT_FOLDER,
    },
    uploadsFolder: process.env.UPLOADS_FOLDER,
    chatGptApiKey: process.env.CHATGPT_API_KEY,
    testing: process.env.NODE_ENV === 'development',
```

```
    port: parseInt(process.env.PORT, 10),
    domain: process.env.DOMAIN,
    protocol: process.env.PROTOCOL,
    brainKnights: {
      id: parsedBrainKnightsProperties.id,
      name: parsedBrainKnightsProperties.name,
      minControllers: parsedBrainKnightsProperties.minControllers,
      maxControllers: parsedBrainKnightsProperties.maxControllers,
      avatars: parsedBrainKnightsProperties.avatars,
      topics: parsedBrainKnightsProperties.topics,
    },
    skibidyParty: {
      id: parsedSkibidyPartyProperties.id,
      name: parsedSkibidyPartyProperties.name,
      minControllers: parsedSkibidyPartyProperties.minControllers,
      maxControllers: parsedSkibidyPartyProperties.maxControllers,
      content: parsedSkibidyPartyProperties.content,
    },
    turingTest: {
      id: parsedTuringTestProperties.id,
      name: parsedTuringTestProperties.name,
      minControllers: parsedTuringTestProperties.minControllers,
      maxControllers: parsedTuringTestProperties.maxControllers,
    },
  };
```

Ці конфігураційні змінні застосовуються на рівні всього додатку. Наприклад, щоб завантажити файли зображень мемів, було створено власний інтерсептор, метод якого перевіряє, чи є користувач, який завантажує файли, організатором ігрової сесії. Якщо це так, він отримує конфігураційні змінні для директорії завантаження файлів та використовує ці дані для налаштування проміжного програмного забезпечення multer, яке використовується для завантаження файлів:

```
  intercept(context: ExecutionContext, next: CallHandler):
Observable<any> {
    const req = context.switchToHttp().getRequest<Request>();
    const { socketid } = req.headers;
    if (typeof socketid !== 'string') {
      throw new NotFoundException(`SocketID is not a valid string.`);
    }
    const room = this.roomService.findRoomByHostSocketId(socketid);
    const isSkibidyPartyGame =
      room?.getGame().id === GameIdEnum.SKIBIDYPARTY;
    if (!room || !isSkibidyPartyGame) {
      throw new NotFoundException(
        `Room with a host socketID ${socketid} was not found.`,
      );
    }
    const { content } =
      this.configService.get<SkibidyPartyGame>('skibidyParty');
    const { memes } = content;
    const { maxUploadedFiles, uploadedFilesFieldName } = memes;
    const roomId = room.getId();
    const localOptions: MulterOptions = {
```

```
      storage: this.getMulterStorage(roomId),
      fileFilter: this.getFileFilter(),
    };
    this.filesInterceptor = new (NestJSFilesInterceptor(
      uploadedFilesFieldName,
      maxUploadedFiles,
      localOptions,
    ))();
    return this.filesInterceptor.intercept(context, next);
  }
```

Для генерації унікальних назв файлів (рис. 4.6) при завантаженні використовується бібліотека crypto:

```
import crypto from 'crypto';
export const fileUniqueSuffix = (): string => {
  return crypto.randomBytes(16).toString('hex');
};
```

В цьому коді генерується рядок, представлений шістнадцятковими цифрами, довжиною в 32 символи, що відповідає 16 байтам.

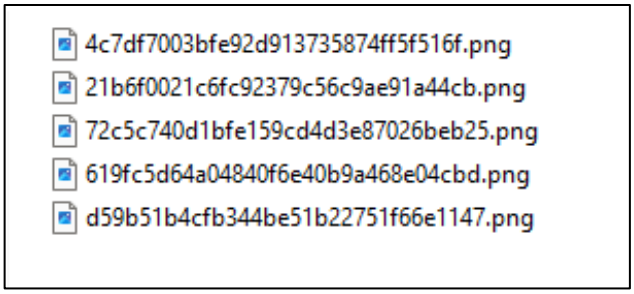


Рисунок 4.6 – Назви збережених файлів зображень для гри Skibidy Party (знімок екрана виконано самостійно)

Наступний фрагмент програмного коду описує логіку призупинення ігрової сесії:

```
public pauseAllTimeouts(): void {
  this.paused = true;
  this.timeouts.forEach((timeoutData) => {
    clearTimeout(timeoutData.timeoutId);
    const timeElapsed = Date.now() - timeoutData.startTime;
    timeoutData.remaining = Math.max(
      0,
      timeoutData.remaining - timeElapsed,
    );
    if (timeoutData.preCompletionTimeoutId) {
      clearTimeout(timeoutData.preCompletionTimeoutId);
    }
  });
}
```

У методі pauseAllTimeouts() спочатку встановлюється змінна paused в true, що сигналізує про призупинення ігрової сесії. Потім він ітерується по масиву timeouts, де для кожного елемента викликається метод clearTimeout, щоб скасувати таймаут. Після цього обчислюється час, який пройшов з моменту початку таймауту до призупинення, і оновлюється значення часу, що залишився до його завершення. Якщо для таймауту встановлено preCompletionTimeoutId, цей таймаут також скасовується. Таким чином, метод призупиняє та зупиняє всі активні таймаути для конкретної ігрової сесії, яка ідентифікується за її ідентифікатором.

Наступний фрагмент програмного коду описує логіку відновлення ігрової сесії:

```
public resumeAllTimeouts(): void {
  this.paused = false;
  this.timeouts.forEach((timeoutData, key) => {
    if (timeoutData.remaining > 0) {
      timeoutData.timeoutId = setTimeout(() => {
        timeoutData.timeoutCallback();
        this.clearTimeout(key);
      }, timeoutData.remaining);
      timeoutData.startTime = Date.now();
      if (timeoutData.preCompletionTimeoutId) {
        const newRemainingpreCompletionTimeout = Math.max(
          0,
          timeoutData.remaining - timeoutData.preCompletionTime,
        );
        timeoutData.preCompletionTimeoutId = setTimeout(() => {
          timeoutData.preCompletionCallback();
          clearTimeout(timeoutData.preCompletionTimeoutId);
        }, newRemainingpreCompletionTimeout);
      }
    }
  });
}
```

Метод resumeAllTimeouts() спочатку встановлює змінну paused в false, що сигналізує про відновлення ігрової сесії. Потім він ітерується по масиву timeouts, для кожного елемента перевіряючи, чи залишений час до завершення таймауту більше 0. Якщо так, то встановлюється новий таймаут з викликом відповідної функції через залишений час. При цьому оновлюється startTime на поточний момент часу. Якщо для таймауту встановлено preCompletionTimeoutId, обчислюється нове значення залишкового часу для цього таймауту і встановлюється новий таймаут для виклику відповідної функції перед

завершенням. Якщо залишений час до завершення таймауту менше або дорівнює 0, таймаут не відновлюється. Крім того, якщо для таймауту встановлено preCompletionTimeoutId і він неактивний (залишений час менше або дорівнює 0), він також скасовується. Таким чином, метод resumeAllTimeouts() відновлює та запускає всі призупинені таймаути для конкретної ігрової сесії, ідентифікуємої за її ідентифікатором.

У розділі 3 на рисунку 3.17 наведена UML-діаграма діяльності отримання повідомлення про призначення мему місця в грі. Фрагменти коду програмної реалізації алгоритму, що описаний в цій діаграмі, також поділено на 5 частин, як і діаграму у розділі 3 на рисунках 3.18-3.22.

Наступний фрагмент коду є частиною реалізації методу assignWinningSpots, який відповідає за призначення місця мему. Цей фрагмент є програмною реалізацією алгоритму, описаного у розділі 3 на рисунках 3.18-3.19, і він реалізує отримання статусу гри, отримання ролі гравця, отримання ситуації, а також перевірку статусу гри, що знаходиться в статусі «розпочато», перевірку того, чи гравець має роль судді, і чи ситуація в межах цієї ігрової сесії містить мем, якому присвоюється місце:

```
public assignWinningSpots(
    controllerId: string,
    memeId: number,
  ): SkibidyPartyMeme | undefined {
    const gameSession = this.getGameSession();
    if (gameSession.status !== GameSessionStatusEnum.STARTED) {
       return undefined;
    }
    const controller = this.findControllerById(controllerId);
    if (controller.actor !== ActorsEnum.JUDGE) {
      return undefined;
    }
    const situation = gameSession.situations.find(
      (situation) => situation.nowDisplayed && !situation.wasUsed,
    );
    if (!situation || !situation.memes.includes(memeId)) {
      return undefined;
    }
    const situationMemesIds = situation.memes;
```

Наступний фрагмент коду є програмною реалізацією алгоритму, описаного у розділі 3 на рисунку 3.20:

```
    if (situationMemesIds.length > 0) {
      const meme = gameSession.memes.find((meme) => meme.id ===
```

```
memeId);
  if (meme && situation.memes.includes(meme.id)) {
    if (!controller.lastSelectedSpot) {
      meme.spot = 1;
      controller.lastSelectedSpot = 1;
    } else if (controller.lastSelectedSpot === 1) {
      const oldMeme = gameSession.memes.find((m) => m.spot === 2);
      if (oldMeme) {
        delete oldMeme.spot;
        if (oldMeme.id === memeId) {
          const oldMeme2 = gameSession.memes.find(
            (m) => m.spot === 3,
          );
          if (oldMeme2) {
            delete oldMeme2.spot;
          }
          meme.spot = 3;
          controller.lastSelectedSpot = 3;
        } else {
          meme.spot = 2;
          controller.lastSelectedSpot = 2;
        }
      } else {
        meme.spot = 2;
        controller.lastSelectedSpot = 2;
      }
    }
```

Наступний фрагмент коду є програмною реалізацією алгоритму, описаного у розділі 3 на рисунку 3.21:

```
    else if (controller.lastSelectedSpot === 2) {
      const oldMeme = gameSession.memes.find((m) => m.spot === 3);
      if (oldMeme) {
        delete oldMeme.spot;
        if (oldMeme.id === memeId) {
          const oldMeme2 = gameSession.memes.find(
            (m) => m.spot === 1,
          );
          if (oldMeme2) {
            delete oldMeme2.spot;
          }
          meme.spot = 1;
          controller.lastSelectedSpot = 1;
        }else {
          meme.spot = 3;
          controller.lastSelectedSpot = 3;
        }
      } else {
        meme.spot = 3;
        controller.lastSelectedSpot = 3;
      }
    }
```

Наступний фрагмент коду є програмною реалізацією алгоритму, описаного у розділі 3 на рисунку 3.22:

```
      else if (controller.lastSelectedSpot === 3) {
        const oldMeme = gameSession.memes.find((m) => m.spot === 1);
        if (oldMeme) {
          delete oldMeme.spot;
          if (oldMeme.id === memeId) {
            const oldMeme2 = gameSession.memes.find(
               (m) => m.spot === 2,
            );
            if (oldMeme2) {
              delete oldMeme2.spot;
            }
            meme.spot = 2;
            controller.lastSelectedSpot = 2;
          } else {
            meme.spot = 1;
            controller.lastSelectedSpot = 1;
          }
        } else {
          meme.spot = 1;
          controller.lastSelectedSpot = 1;
          }
      }
    }
  return meme;
  }
  return undefined;
}
```

Наступний фрагмент коду описує метод, який відповідає за розподіл гравців гри Turing Test по командам:

```
public assignControllerToTeam(): TeamEnum {

  let team = TeamEnum.STUDENTS;
  const controllersPerTeamInRoom =
this.countControllersPerTeamInRoom();
  const studentsCount = controllersPerTeamInRoom.students;
  const professorsCount = controllersPerTeamInRoom.professors;
  if (studentsCount === 1 && professorsCount < 4) {
    team = TeamEnum.PROFESSORS;
  } else if (studentsCount === 1 && professorsCount === 4) {
    team = TeamEnum.STUDENTS;
  } else if (studentsCount === 2 && professorsCount < 5) {
    team = TeamEnum.PROFESSORS;
  } else if (studentsCount === 2 && professorsCount === 5) {
    team = TeamEnum.STUDENTS;
  }
  return team;
}
```

Розподіл відбувається згідно з наступним співвідношенням: 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.

Також на back-end частині було здійснено налаштування мережевої гри. Для цього для кожного класу-шлюзу, який обробляє WebSocket повідомлення, було додано наступний декоратор:

```
@WebSocketGateway({
  namespace: '/turingtest',
  cors: {
    origin: `${process.env.PROTOCOL}://${process.env.DOMAIN}`,
  },
})
```

Цей декоратор використовується для налаштування WebSocket шлюзу у NestJS. Параметр namespace визначає простір імен для WebSocket, у цьому випадку – «/turingtest». Це означає, що всі WebSocket з'єднання, які використовуються з цим шлюзом, будуть робитися в просторі імен «/turingtest». Назва простору імен відображає назву гри для якої призначений цей шлюз, щоб ідентифікувати її при взаємодії з клієнтами. Параметр cors визначає політику CORS для дозволу з'єднання з певного джерела.

Крім того, всередині класу було створено сервер WebSocket, який буде приймати з'єднання від клієнтів:

```
@WebSocketServer() server: Server = new Server<
  ServerToClientEvents,
  ClientToServerEvents
>();
```

У клієнтській частині, що використовує React та JavaScript, було використано бібліотеку socket.io-client для забезпечення мережевої взаємодії через протокол WebSocket з back-end частиною. Це дозволяє обмінюватися повідомленнями та даними з сервером для їх подальшої обробки й відправки повідомлень іншим клієнтам, що забезпечує мережеву взаємодію між ними. Наступний код на JavaScript створює підключення до WebSocket-сервера за допомогою бібліотеки Socket.IO:

```
const target = process.env.REACT_APP_BACKEND_URL
const socket = io(`${target}/turingtest`, {
  autoConnect: false,
})
```

Наступний код використовує React хук useEffect для керування WebSocket-з'єднанням за допомогою бібліотеки Socket.IO:

```
useEffect(() => {
  socket.on('connect', handleConnect)
```

```
    socket.on('disconnect', handleDisconnect)
    socket.connect()
    return () => {
      socket.disconnect()
      socket.off('connect', handleConnect)
      socket.off('disconnect', handleDisconnect)
    }
  }, [
    handleConnect,
    handleDisconnect,
  ])
```

Цей код ініціалізує підключення до сервера, встановлює обробники подій для повідомлень від сервера, таких як підключення та відключення. Підключення до сервера виконується методом socket.connect(), що дозволяє отримувати дані в реальному часі, необхідні для роботи застосунку. При демонтажі компонента або зміні залежностей, цей код виконує очистку, викликаючи socket.disconnect() для розриву з’єднання та видаляючи всі обробники подій за допомогою socket.off. Це забезпечує коректне управління ресурсами та запобігає витокам пам’яті, гарантуючи, що підключення буде правильно закрите, а обробники подій не залишаться прив’язаними після того, як компонент більше не використовується.

Цей код дозволяє клієнтській стороні взаємодіяти з back-end частиною, яка надає необхідний функціонал для мережевої гри.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування повинно охоплювати всі функції системи, включаючи створення і управління гральними кімнатами, синхронізацію гри між усіма гравцями, обробку дій гравців, обробку та збереження даних сесії гри протягом часу їх існування.

З огляду на невеликий розмір проєкту та готовність клієнтської частини до взаємодії з back-end частиною, було обрано ручне тестування. Цей підхід дозволяє приділити більше уваги деталям та виявити найбільш критичні проблеми. Він також допоможе швидше виявляти та виправляти помилки, оскільки ручне тестування дозволяє тестувальнику безпосередньо взаємодіяти з системою та оперативно реагувати на виявлені проблеми.

Крім того, ручне тестування дозволяє застосовувати творчий підхід до виявлення проблем і вразливостей, що особливо корисно при перевірці складних функцій або взаємодії різних компонентів системи.

Для ручного тестування цього ігрового програмного застосунку потрібно використовувати структурований підхід, щоб охопити всі функціональні та нефункціональні вимоги, зазначені у специфікації. Процес тестування ігрового програмного застосунку включає наступні етапи:

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

б) перевірка функціональних вимог:

1) для гри Skibidy Party:

– перевірено, що організатор може завантажувати власні файли зображень мемів;

– перевірено, що виконується валідація файлів, які завантажуються;

– перевірено, що створюється колода карток з урахуванням завантажених та системних мемів;

– перевірено, що після того, як гравець увійшов в кімнату, де ігрова сесія ще не розпочалася, то йому автоматично надається випадковий аватар;

– перевірено, що гравець може змінювати аватар;

– перевірено, що перед початком кожного раунду випадковому гравцеві, який ще не був суддею, надається роль судді, і іншим – роль звичайного гравця;

– перевірено, що користувач отримує файли зображень мемів;

– перевірено, що обирається випадковим чином ситуація, яка ще не була обрана раніше;

– перевірено, що організатор отримує випадково обрану ситуацію для раунду;

– перевірено, що картки розподіляються між звичайними гравцями, з виключенням тих карток, які вже були розподілені;

– перевірено, що звичайний гравець може надсилати свій вибір найсмішнішого мема;

– перевірено, що суддя отримує інформацію про вибір найсмішніших мемів за думкою звичайних гравців;

– перевірено, що суддя може надсилати дані про призначені мемам місця від першого до третього;

– перевірено, що бали нараховуються звичайним гравцям в залежності від місця, на яке поставив їх мем суддя;

– перевірено, що організатор отримує інформацію про бали гравців з попереднього раунду та поточного;

– перевірено, що користувач отримує дані про переможців раунду та гри;

2) для гри Turing Test:

– перевірено, що гравці розподіляються 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;

– перевірено, що гравець може змінити команду відповідно до правил, якщо є вільні місця;

– перевірено, що гравець з команди студентів може надіслати питання;

– перевірено, що гравець з команди професорів отримує питання від студентів;

– перевірено, що гравець з команди професорів може надіслати відповідь на питання;

– перевірено, що організатор отримує всі відповіді до кожного питання, як від професорів, так і від штучного інтелекту, без вказання, чия це відповідь;

– перевірено, що гравець з команди студентів може надсилати свої вибори відповідей, які, на його думку, надав штучний інтелект;

– перевірено, що організатор отримує відповіді, які обирають гравці з команди студентів;

– перевірено, що відповіді, які обрали гравці з команди студентів обробляються належним чином;

– перевірено, що організатор отримує дані про те, чи помилилися гравці з команди студентів під час вибору штучного інтелекту чи ні;

– перевірено, що бали команді переможців нараховуються;

– перевірено, що користувач отримує дані про нараховані бали, а також дані про переможця, як в кінці кожного раунду так і в кінці гри.

3) для всіх ігор разом:

– перевірено, що організатор може створювати та входити в кімнату;

– перевірено, що гравець може входити в кімнату;

– перевірено, що гравцю, який увійшов першим, привласнюється статус власника кімнати;

– перевірено, що гравець може виходити з кімнати;

– перевірено, що статус власника кімнати автоматично передається наступному в черзі гравцю, який увійшов після попереднього власника кімнати, котрий вийшов з кімнати;

– перевірено, що організатор може змінювати мову в кімнаті;

– перевірено, що організатор може встановлювати мову кімнати загальною для всіх її учасників;

– перевірено, що власник кімнати може розпочинати ігрову сесію для учасників кімнати;

– перевірено, що організатор може ставити ігрову сесію на паузу та відновлювати її роботу;

– перевірено, що організатор може виходити з кімнати;

– перевірено, що після того, як організатор залишає кімнату, відбувається завершення ігрової сесії та видалення її даних;

– перевірено, що користувач отримує дані про кімнату, ігрову сесію в ній та інформацію про себе;

– перевірено, що досягнення розподіляються серед гравців;

– перевірено, що виконується валідація отриманих даних;

– перевірено, що перед початком гри формується черга відображення екранів гри для кожного типу гравця; всі елементи черги повинні містять множину елементів, при цьому кожен елемент цієї множини включає назву екрану, інформацію про те, для кого він призначений, тривалість його відображення та функції, які необхідно виконати під час, перед або після його появи; змінюється черга екранів та її елементи, а також ті елементи, які вже вийшли з неї, при необхідностях, що випливають з логіки гри;

– перевірено, що користувач отримує інформацію про час відображення поточного екрану.

в) перевірка нефункціональних вимог:

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

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

3) перевірено, що файли зображень зберігаються в локальному файловому сховищі на тому самому пристрої, де знаходиться back-end частина.

В результаті проведеного тестування були створені тест-кейси для деяких функціональних вимог, які наведені в додатку Д.

6 ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

NestJS – це фреймворк для створення веб-застосунків на Node.js. Оскільки ігровий програмний застосунок розроблений з використанням цього фреймворку, його можна розгорнути на хмарній платформі. Для цього було обрано Render. Render – це хмарна платформа, яка дозволяє безкоштовно розміщувати статичні веб-сайти та API [12].

Перш ніж розгортати застосунок у Render, у сегменті сценарію package.json була додана команда start:prod (рис. 6.1), призначена для запуску програми у виробничому середовищі.

Рисунок 6.1 – Сегменті сценарію package.json (знімок екрана виконано самостійно)

Після цього на GitHub було створено репозиторій, куди було додано файли програми (рис. 6.2).

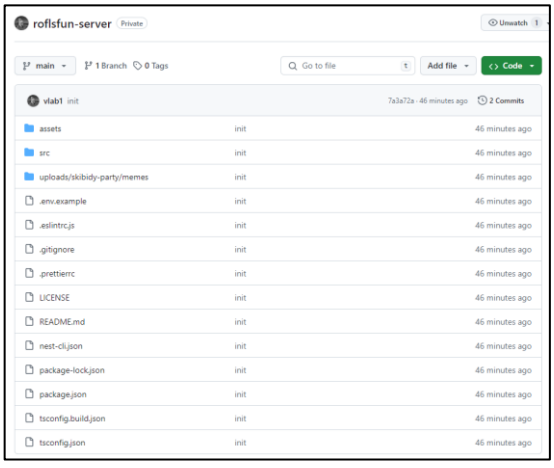


Рисунок 6.2 – Файли GitHub-репозиторію з файлами back-end частини застосунку (знімок екрана виконано самостійно)

Після завершення підготовки до розгортання застосунку на платформі Render було створено безкоштовний веб-сервіс. Для цього було здійснено вхід до Render за допомогою GitHub і надано платформі доступ до створеного репозиторію. Для коректної роботи застосунку було налаштовано команди збірки та запуску застосунку (рис. 6.3).

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

Окрім цього, були додані змінні середовища з вказаними актуальними даними домену клієнтської частини, ключа API OpenAI та мережевого протоколу.

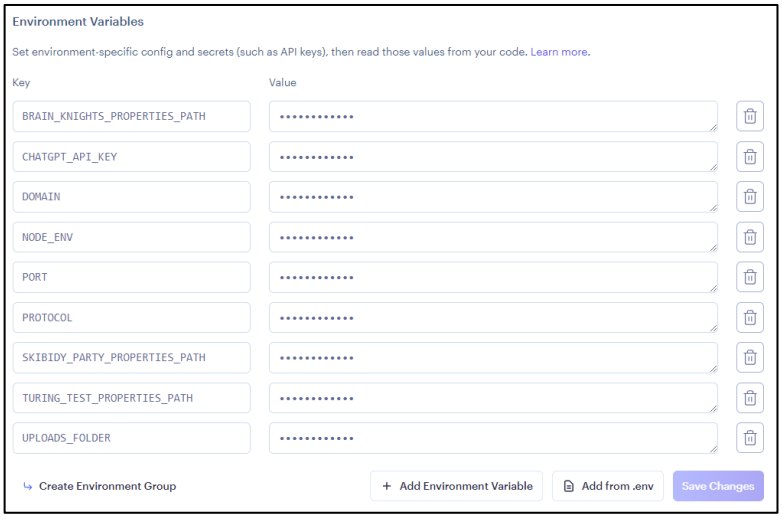


Рисунок 6.4 – Налаштування змінних середовища веб-сервісу на платформі Render (знімок екрана виконано самостійно)

Після завершення налаштувань було розпочато розгортання останнього коміту з GitHub-репозиторію, пов’язаного з веб-сервісом Render.

Коли розгортання завершилося, ігрова сесія гри Skibidy Party була створена (рис. 6.5), використовуючи клієнтську частину застосунку та розгорнутий back-end.

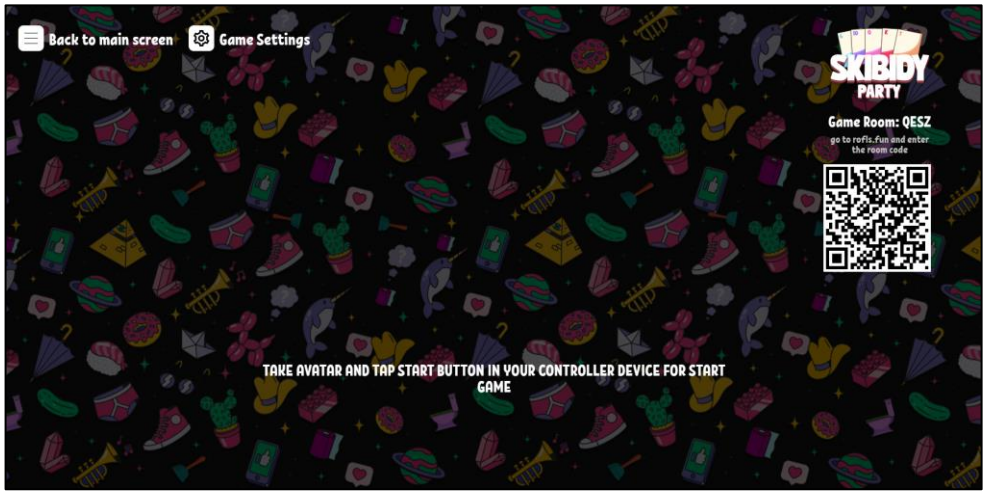


Рисунок 6.4 – Створена ігрова сесія гри Skibidy Party (знімок екрана виконано самостійно)

Далі, за допомогою клієнтської частини, була знайдена кімната цієї ігрової сесії, і до неї було приєднано гравця. Запит з клієнтської частини до розгорнутої back-end частини застосунку було успішно виконано (рис. 6.5).

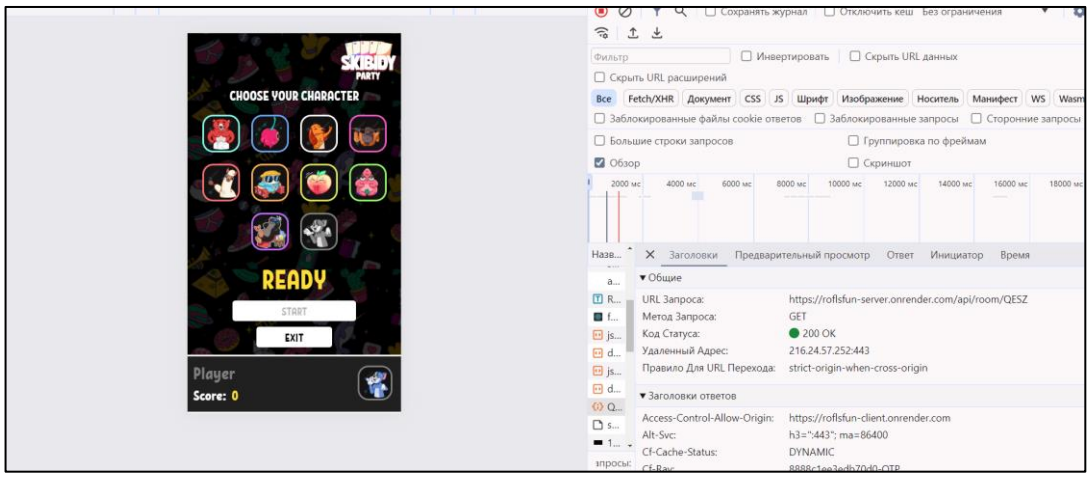


Рисунок 6.5 – Запит з клієнтської частини до розгорнутої back-end частини застосунку було успішно виконано (знімок екрана виконано самостійно)

Це підтверджується також записами у логах (рис. 6.6), де зафіксовано взаємодію клієнтів із back-end частиною.

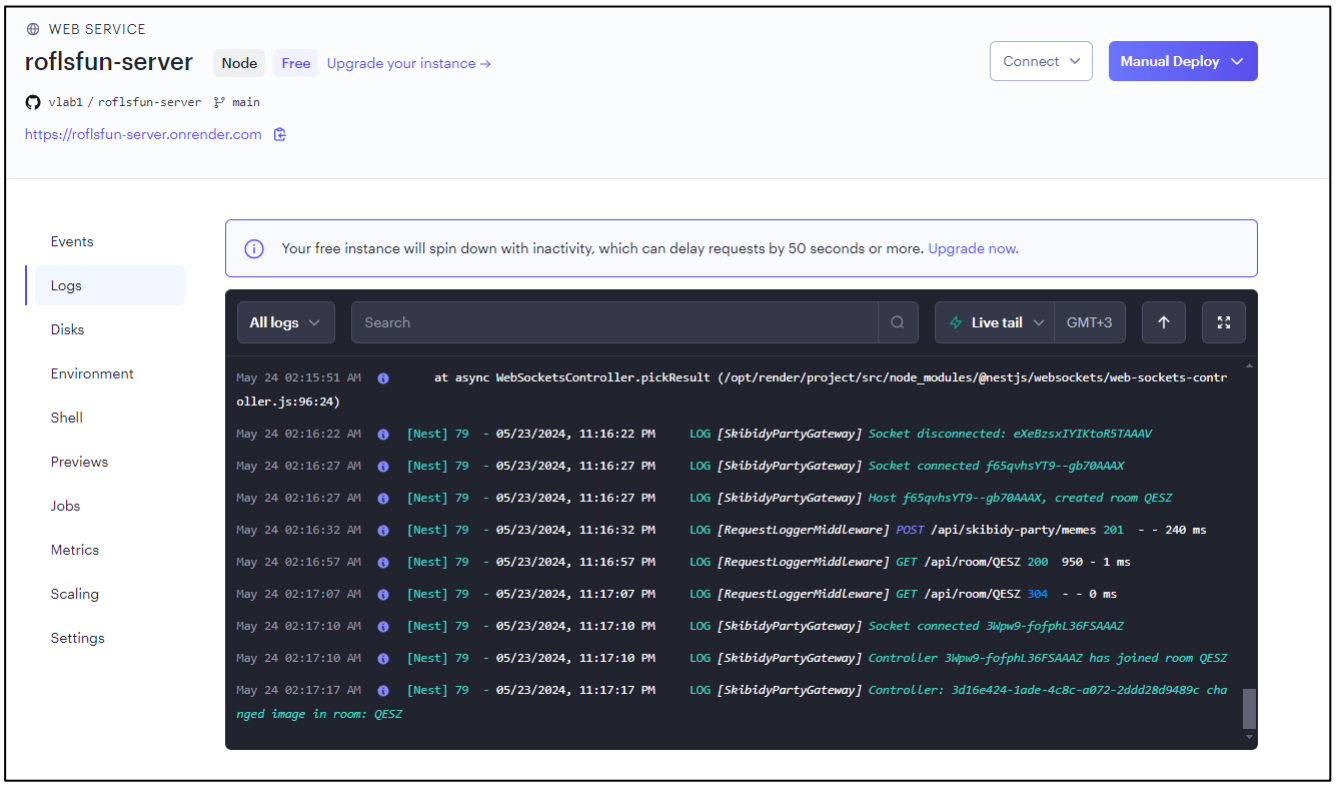


Рисунок 6.6 – Логи ігрового програмного застосунку, розгорнутого на платформі Render (знімок екрана виконано самостійно)

Таким чином, back-end частина розробленого ігрового програмного застосунку була впроваджена в мережу. При використанні клієнтської частини, яка також буде розгорнута у мережі, гра ROFLSFUN стане доступною через веб-браузер.

Ще одним способом впровадження розробленого програмного забезпечення була участь у виставці технічної творчості молоді на 28-му Міжнародному форумі «Радіоелектроніка та молодь у XXI столітті» (додаток Е). На цій виставці я разом із іншими членами команди розробників представив ігровий програмний застосунок у жанрі multiplayer party games, back-end частина якого була розроблена в рамках цієї кваліфікаційної роботи, і ми отримали друге місце (додаток Ж).

Також була взята участь у конференції з теми «Інформаційні інтелектуальні системи» на 28-му Міжнародному форумі «Радіоелектроніка та молодь у XXI столітті» (додаток И). Представлені були тези на тему «Використання структур даних для реалізації черги подій в іграх з механікою обмеження часу». Ця робота стосується розробки модуля для back-end частини ігрового програмного застосунку у жанрі multiplayer party games, який керує формуванням черги відображення екранів гри для різних типів гравців. Кожен елемент черги включає назву екрану, інформацію про те, для кого він призначений, тривалість його відображення та функції, які необхідно виконати перед, під час або після його появи. Додатково, є можливість змінювати чергу екранів та її елементи, а також ті елементи, які вже вийшли з черги, при необхідності, що випливає з логіки гри.

В цілому, розроблений ігровий програмний застосунок ROFLSFUN успішно впроваджено в мережу через веб-браузер. Виставка відзначилася другим місцем, а доповідь на конференції висвітлила технічні аспекти розробленого програмного забезпечення, зокрема, використання структур даних для ефективного управління чергою подій у грі. Всі ці аспекти свідчать про успішну реалізацію проєкту та його потенціал для подальшого розвитку та використання.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було успішно створено back-end частину системи і налаштовано мережеву гру для ігрового програмного застосунку у жанрі multiplayer party games. Розробка back-end частини системи охопила створення логіки двох багатокористувацьких ігор для вечірок наступних жанрів: соціальна дедуктивна гра Turing Test та гумористична карткова гра Skibidy Party.

Перед початком розробки було проведено проєктування архітектури та UML-проєктування back-end частини програмного забезпечення. Це дозволило чітко визначити структуру та взаємозв’язки між різними компонентами системи і, в результаті, сприяло успішній реалізації проєкту.

Під час розробки використовувалися наступні інструменти та технології: редактор Visual Studio Code, платформа Node.js, фреймворк NestJS. Для забезпечення зв’язку між сервером і клієнтами використовувалися WebSocket-сервер та бібліотека Socket.IO. Мови програмування, що використовувалися, – TypeScript і JavaScript.

Основні результати роботи можуть бути узагальнені наступним чином:

– розробка back-end частини системи: в рамках роботи була успішно реалізована back-end частина системи, яка відповідає за керування грою, створення і управління гральними кімнатами, синхронізацію гри між усіма гравцями, обробку дій гравців, обробку та збереження даних сесії гри протягом часу їх існування; ця частина програмного забезпечення забезпечує ефективну та безперебійну роботу ігрового середовища;

– налаштування мережевої гри: під час роботи було проведено налаштування мережевої гри, що дозволяє гравцям підключатися до грального сесії та взаємодіяти один з одним у реальному часі; на клієнтській стороні було реалізовано інтеграцію Socket.IO для налаштування WebSocket-з’єднання з метою забезпечення зв’язку між клієнтами та сервером; на back-end частині було створено

та налаштовано WebSocket-сервер з використанням бібліотеки Socket.IO; використання WebSocket-серверу та бібліотеки Socket.IO забезпечує стабільну та швидку комунікацію між клієнтами та сервером.

Отже, результати роботи свідчать про успішну реалізацію поставлених завдань та теми. Можна зробити висновок, що мета кваліфікаційної роботи була досягнута, а ігровий програмний застосунок у жанрі multiplayer party games готовий до використання. Він готовий для гри через мережу для широкого спектру користувачів, від дорослих до підлітків, під час різних заходів, від онлайн-вечірок з друзями до особистих зустрічей. Це досягнуто, зокрема, завдяки правильно побудованій back-end частині системи та налаштованій мережевій грі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Lombardi A. WebSocket: Lightweight Client-Server Communications. O'Reilly Media, Incorporated, 2015.

2. Introduction | Socket.IO. Socket.IO. URL: https://socket.io/docs/v4/ (дата звернення: 13.05.2024).

3. Vickler A. Javascript: Javascript Front End Programming. Independently Published, 2021.

4. Publishing T. TypeScript Programming Language. Independently Published, 2019. 248 с.

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

6. Index | Node.js v22.1.0 Documentation. Node.js – Run JavaScript Everywhere. URL: https://nodejs.org/docs/latest/api/ (дата звернення: 13.05.2024).

7. Documentation | NestJS – A progressive Node.js framework. Documentation | NestJS – A progressive Node.js framework. URL: https://docs.nestjs.com/ (дата звернення: 13.05.2024).

8. Duldulao D. B., Cabagnot R. J. L. Practical Enterprise React. Berkeley, CA : Apress, 2021. URL: https://doi.org/10.1007/978-1-4842-6975-6 (дата звернення: 13.05.2024).

9. Lanjewar R. Build Scalable and Reliable Node.js Apps with Ease: Introducing Nest JS / Blogs / Perficient. Perficient Blogs. URL: https://blogs.perficient.com/2024/01/30/build-scalable-and-reliable-node-js-apps-with-ease-introducing-nestjs/ (дата звернення: 13.05.2024).

10. Бухало В. О. Використання структур даних для реалізації черги подій в іграх з механікою обмеження часу / В. О. Бухало // Радіоелектроніка та молодь у XXI столітті: тези доповідей 28-го Міжнародного молодіжного форуму, 16–18 квітня 2024 р. – Харків : ХНУРЕ, 2024. – Т. 6. – С. 375–377.

11. 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.

12. Ojiyi C. The Complete Guide to Deploying NestJS Application on Render | HackerNoon. HackerNoon - read, write and learn about any technology. URL: https://hackernoon.com/the-complete-guide-to-deploying-nestjs-application-on-render (дата звернення: 24.05.2024).

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

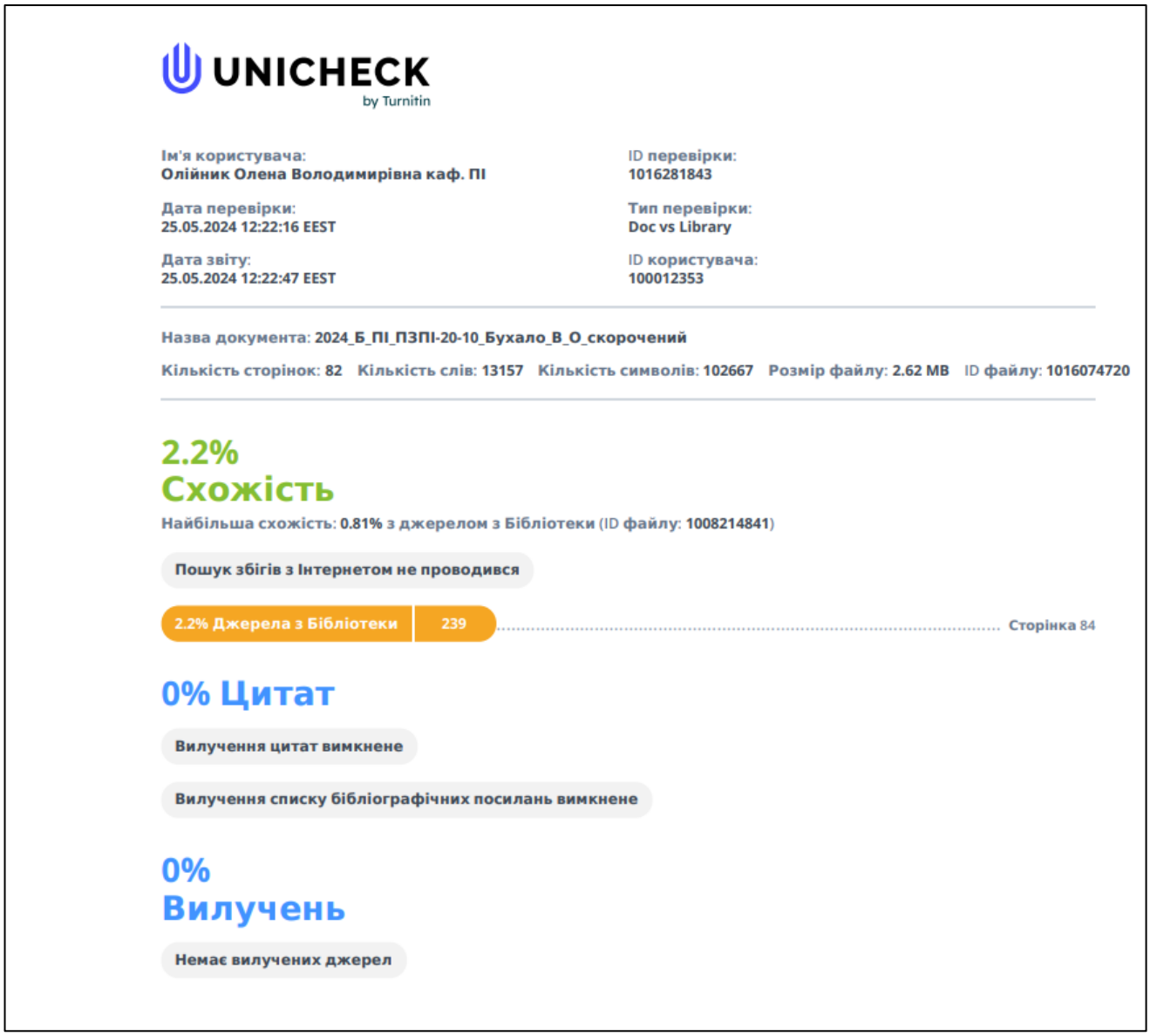


Рисунок А.1 – Результат перевірки на унікальність тексту в базі ХНУРЕ (знімок екрана виконано самостійно)

ДОДАТОК Б

Слайди презентації

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

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

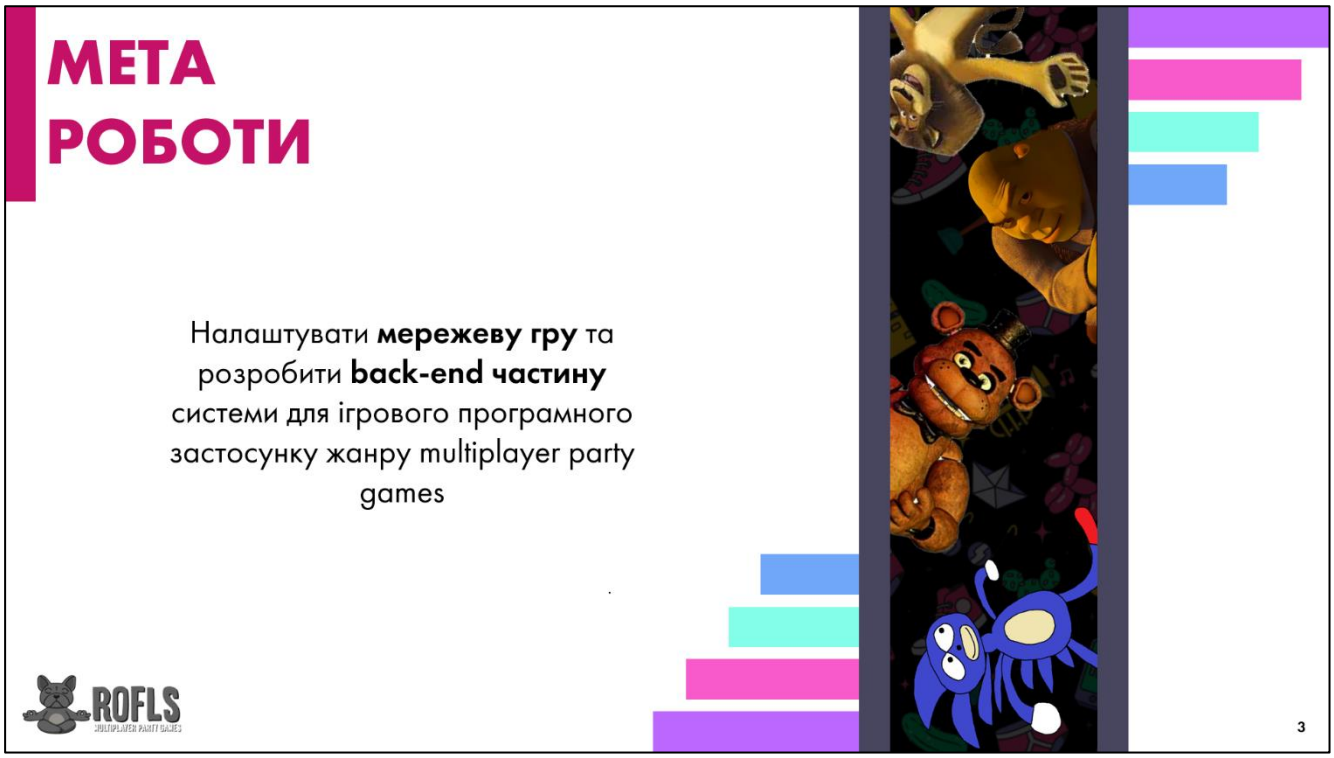


Рисунок Б.3 – Третій слайд презентації (знімок екрана виконано самостійно)

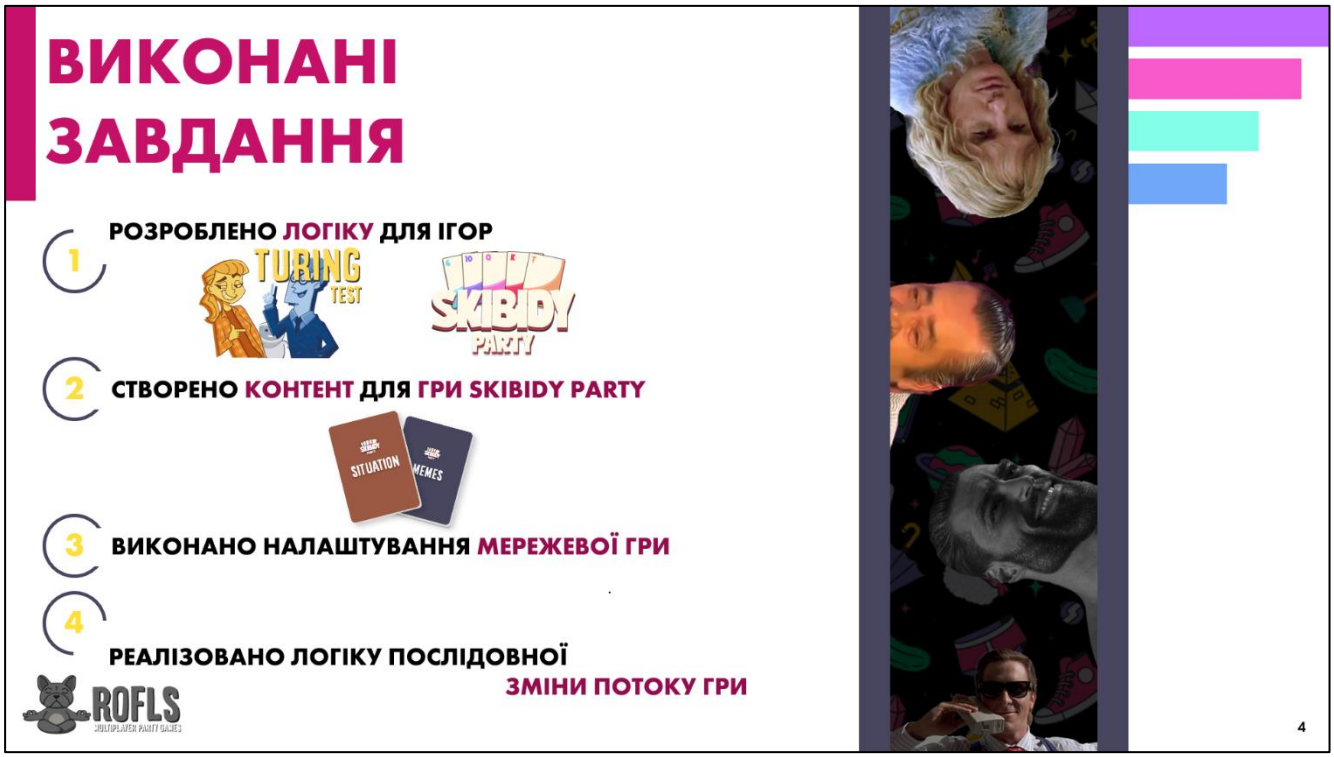


Рисунок Б.4 – Четвертий слайд презентації (знімок екрана виконано самостійно)

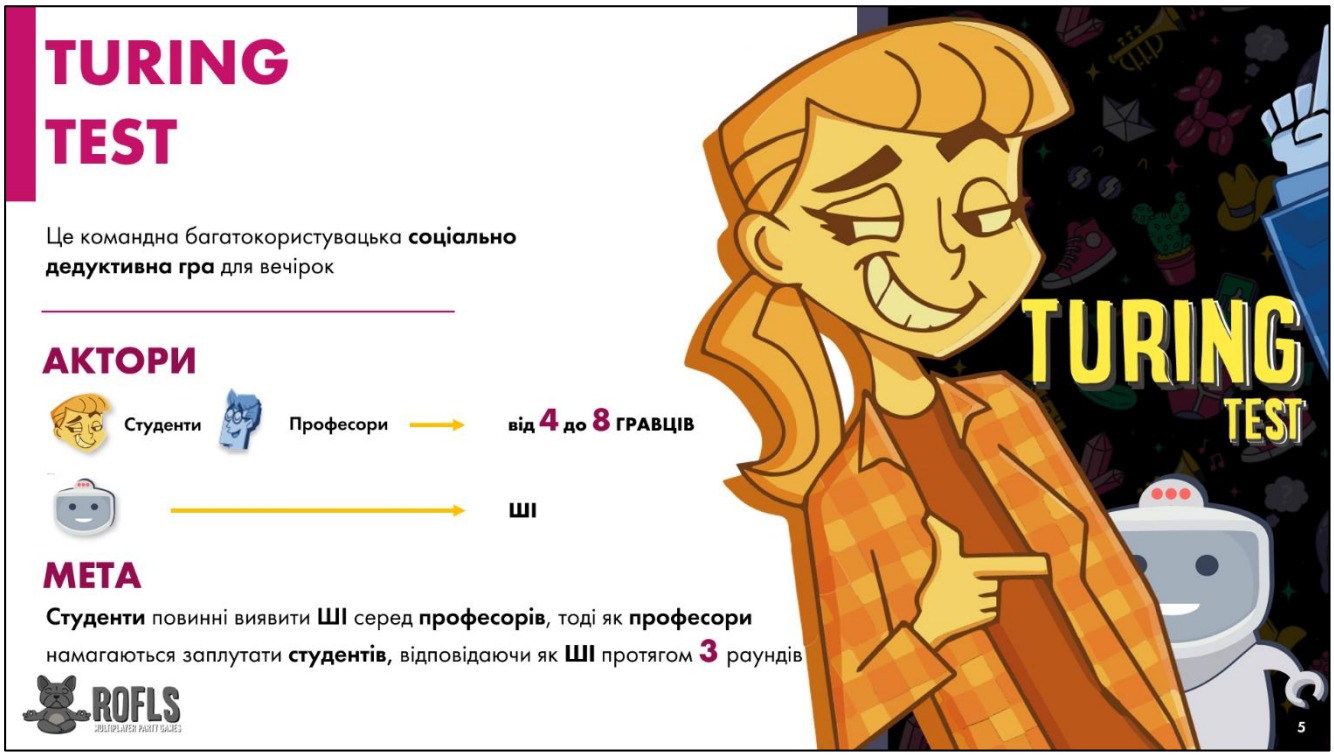


Рисунок Б.5 – П’ятий слайд презентації (знімок екрана виконано самостійно)

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

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

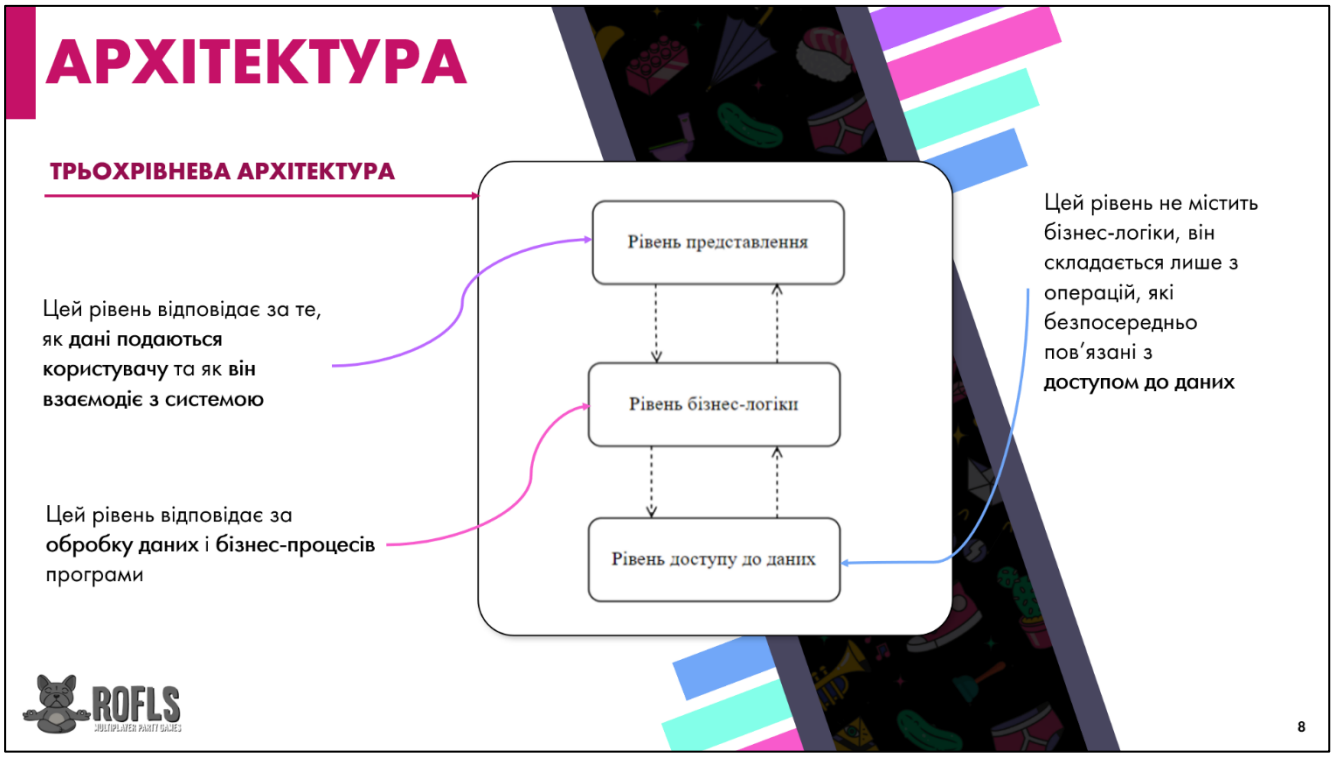


Рисунок Б.8 – Восьмий слайд презентації (знімок екрана виконано самостійно)

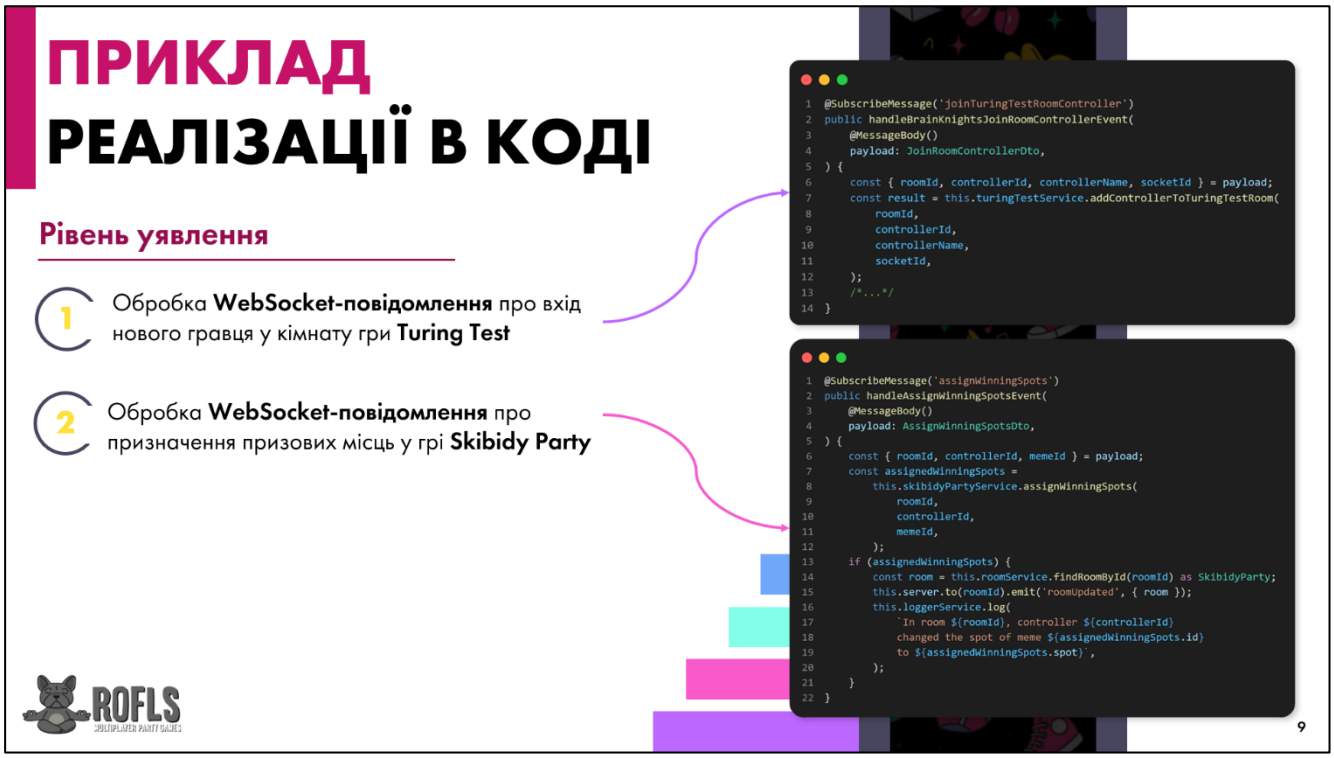


Рисунок Б.9 – Дев’ятий слайд презентації (знімок екрана виконано самостійно)

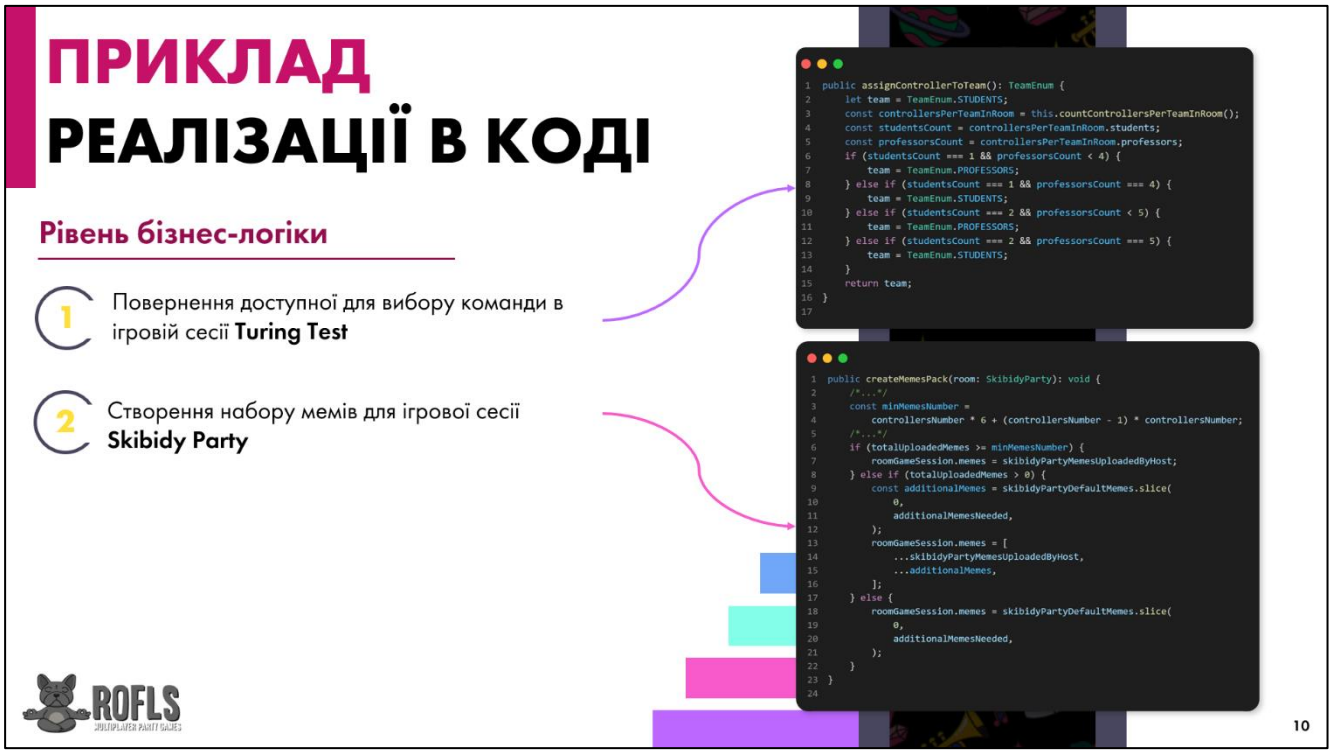


Рисунок Б.10 – Десятий слайд презентації (знімок екрана виконано самостійно)

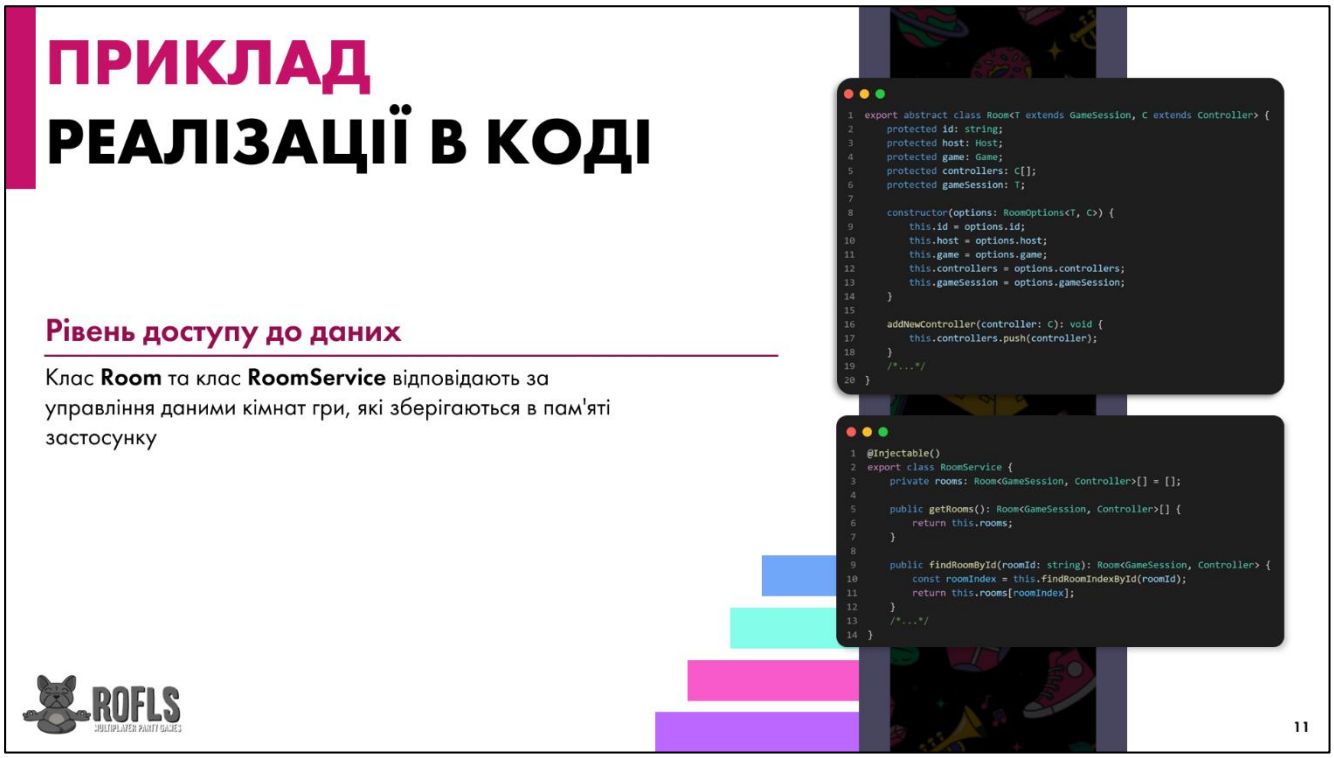


Рисунок Б.11 – Одинадцятий слайд презентації (знімок екрана виконано самостійно)

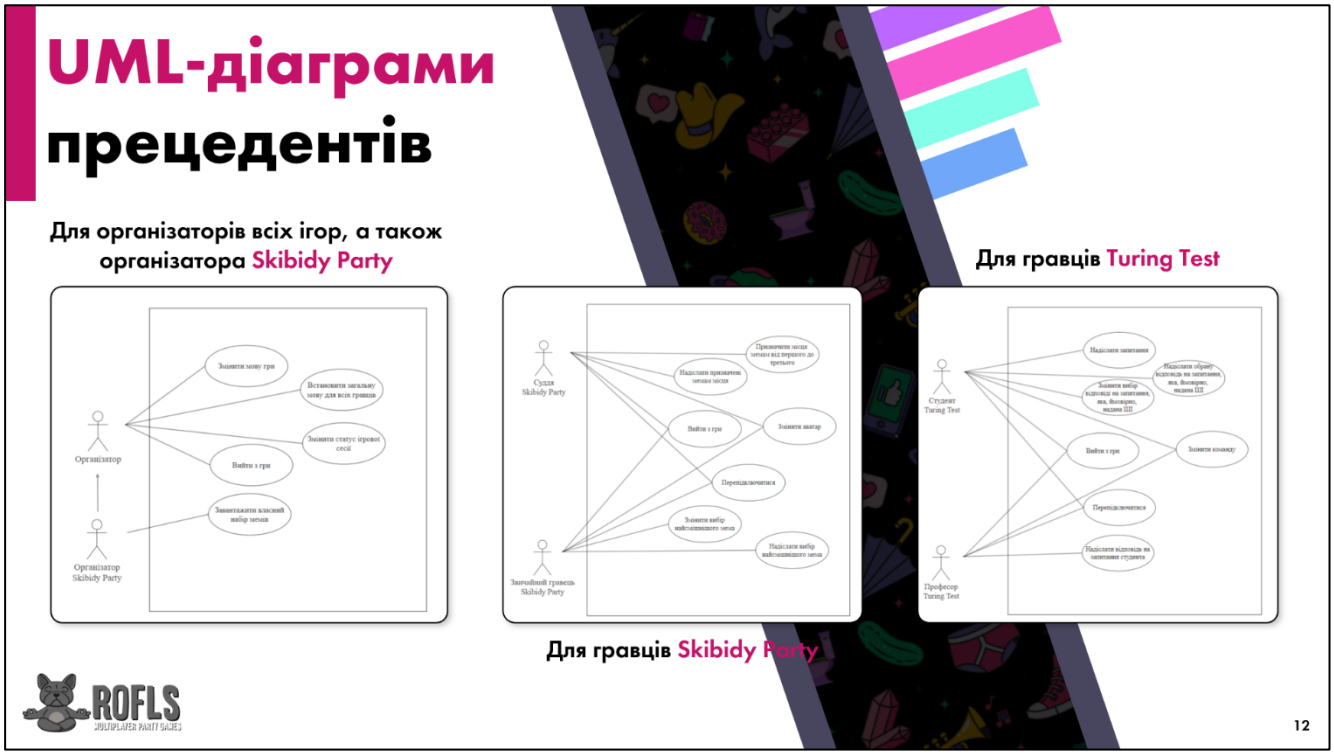


Рисунок Б.12 – Дванадцятий слайд презентації (знімок екрана виконано самостійно)

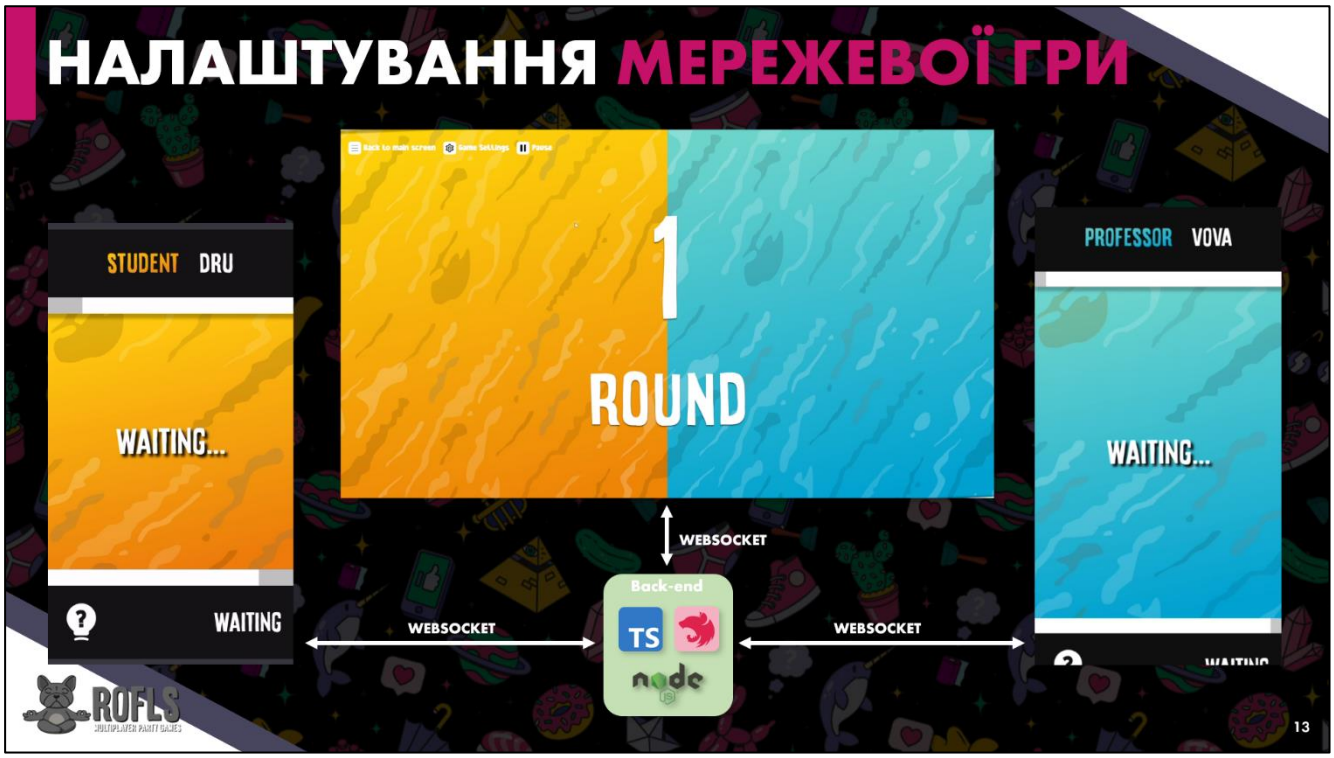


Рисунок Б.13 – Тринадцятий слайд презентації (знімок екрана виконано самостійно)

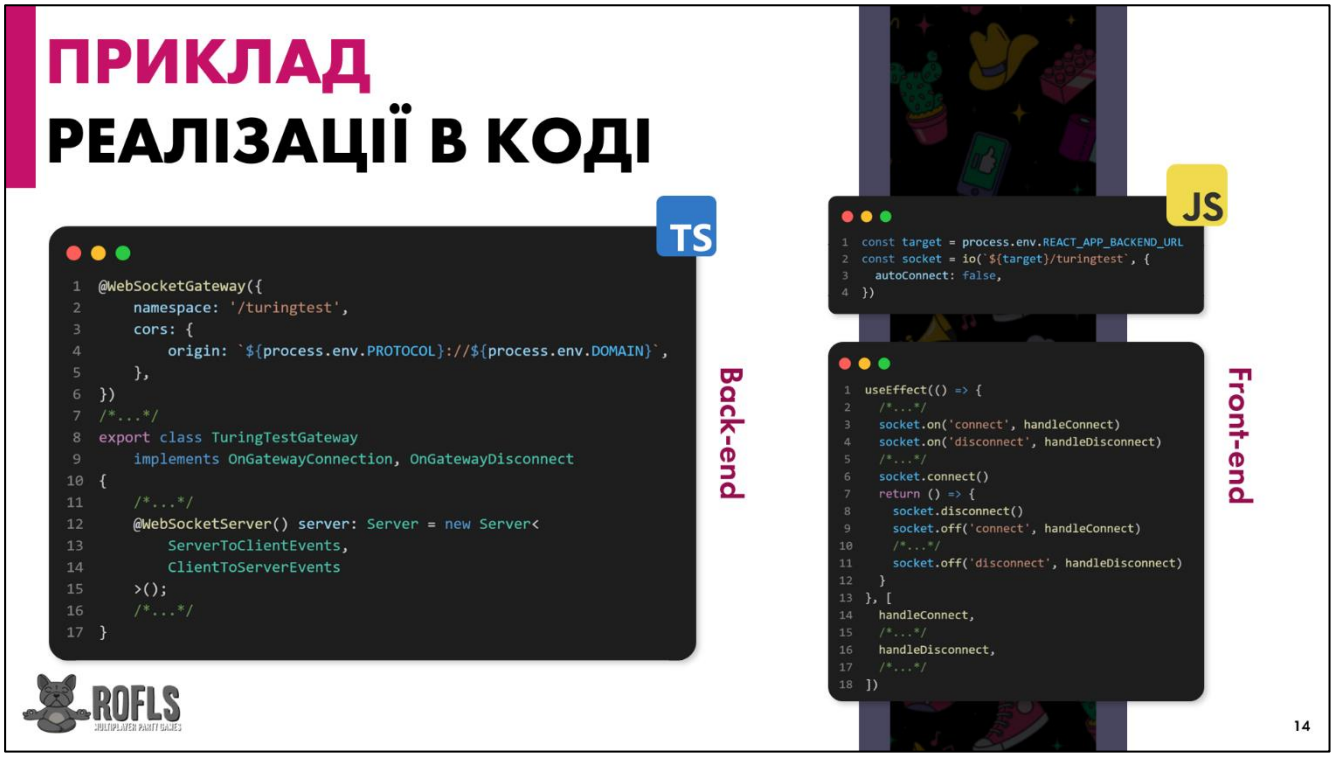


Рисунок Б.14 – Чотирнадцятий слайд презентації (знімок екрана виконано самостійно)

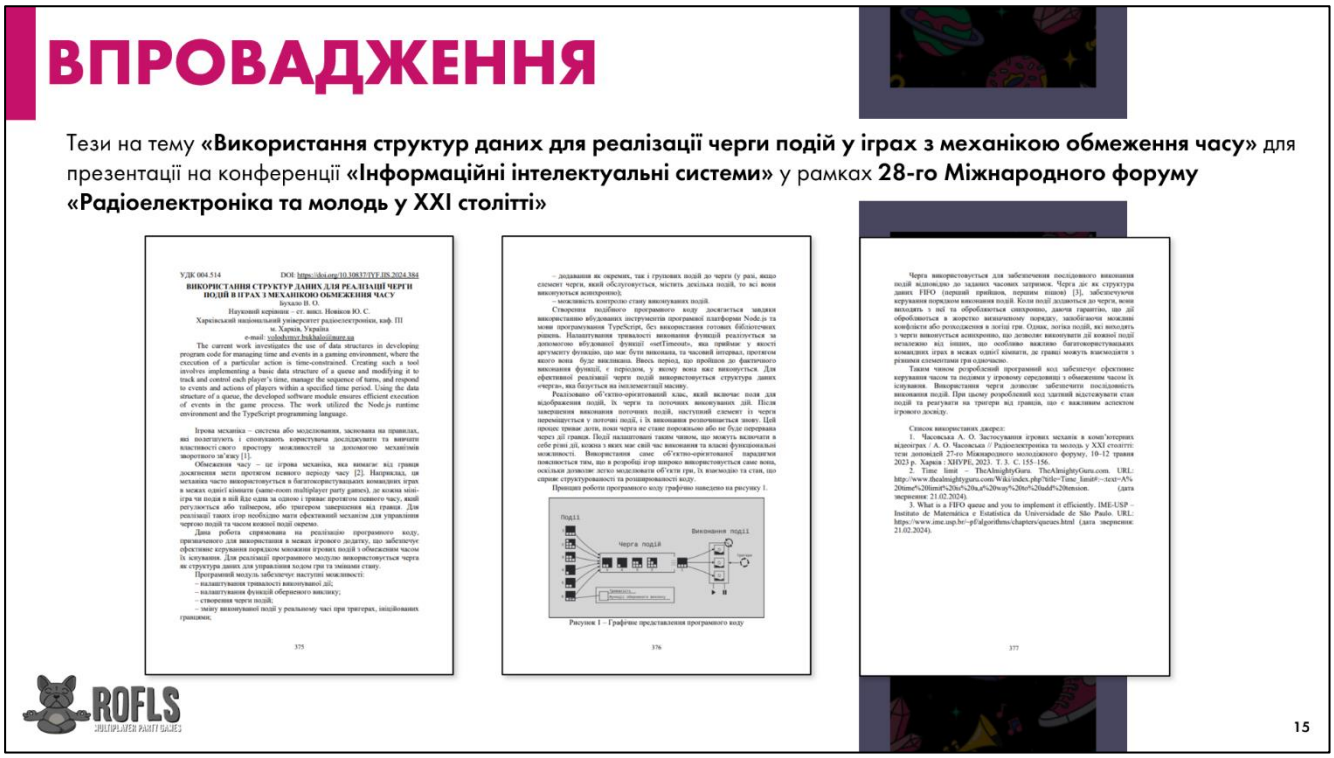


Рисунок Б.15 – П’ятнадцятий слайд презентації (знімок екрана виконано самостійно)

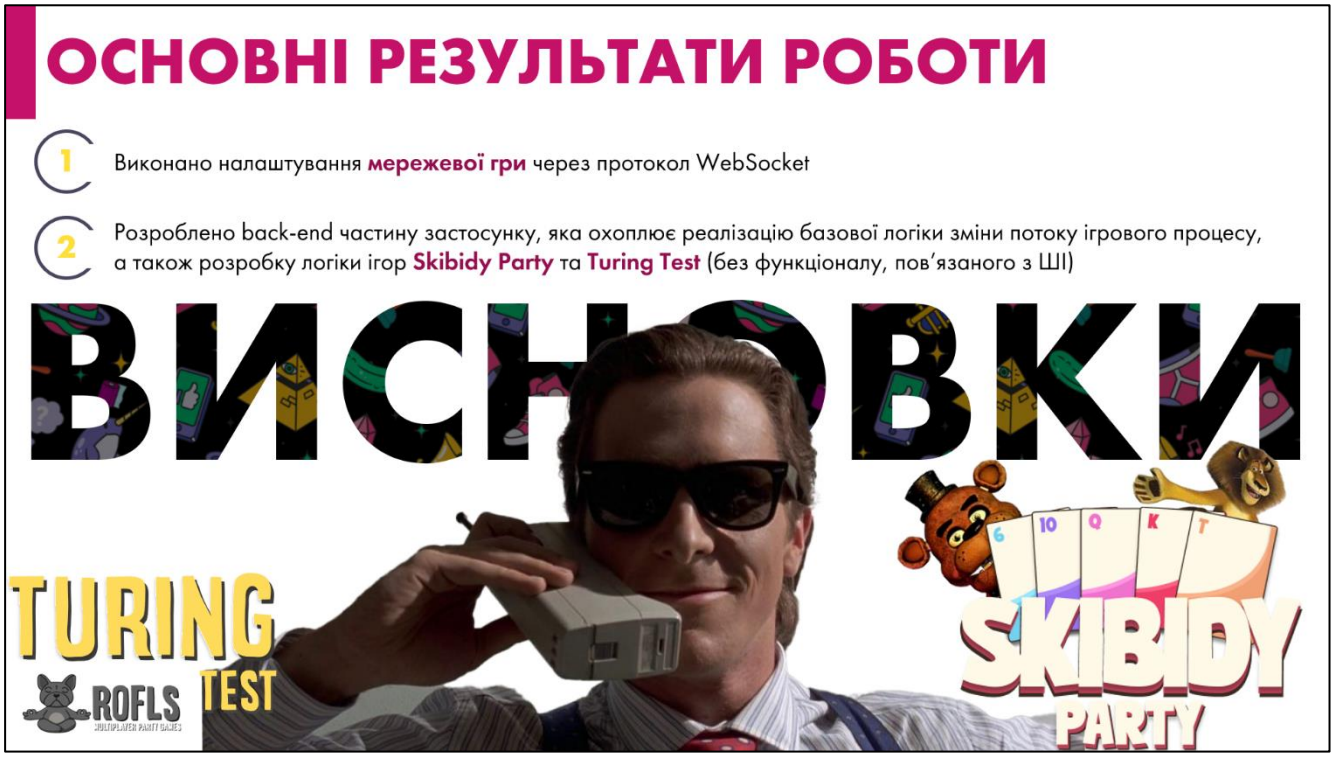


Рисунок Б.16 – Шістнадцятий слайд презентації (знімок екрана виконано самостійно)

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

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.

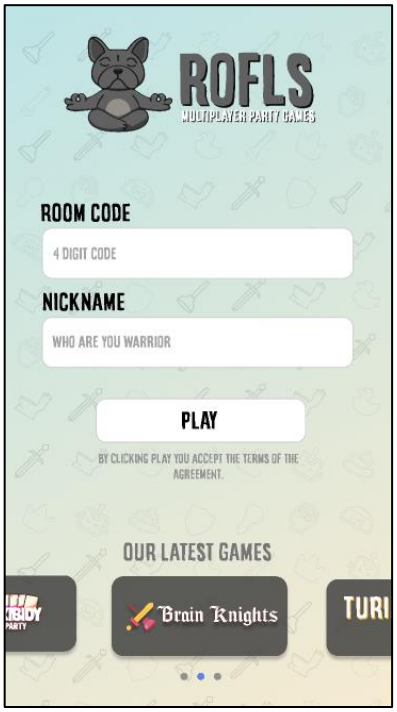


Рисунок 3.2 – Інтерфейс стартового екрану контролера (знімок екрана виконано самостійно)

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.

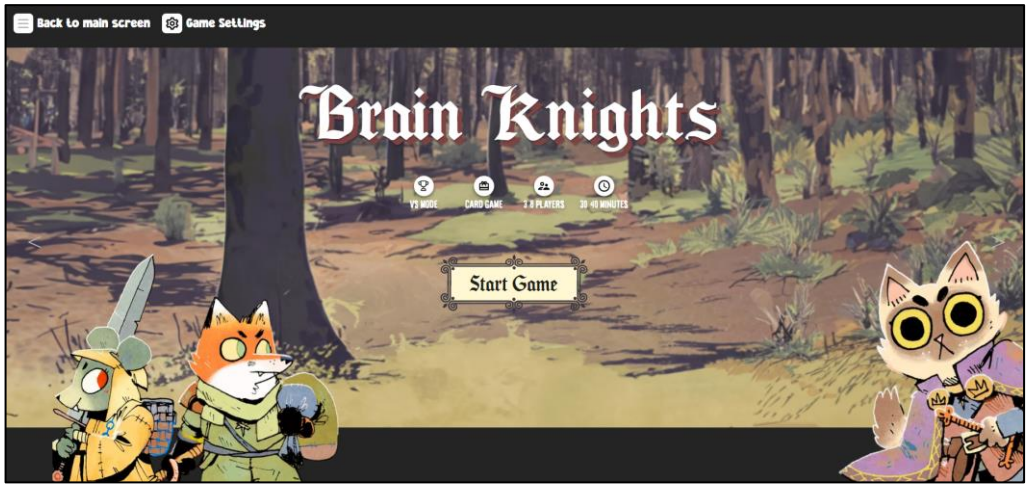


Рисунок 3.3 – Вибір гри Brain Knights (знімок екрана виконано самостійно)

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

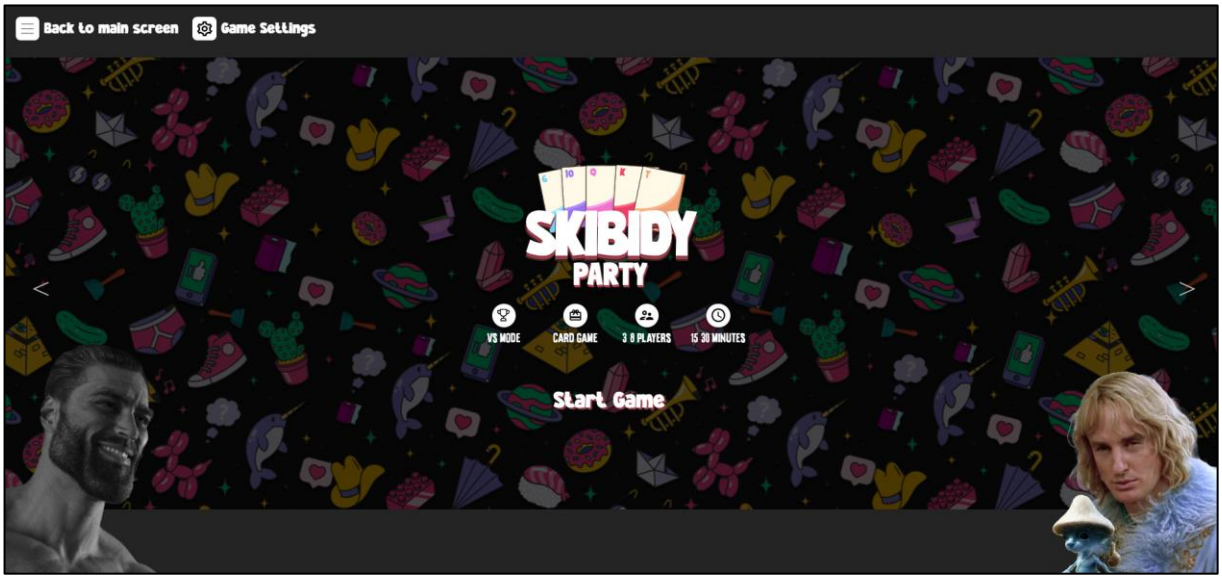


Рисунок 3.4 – Вибір гри Skibidy Party (знімок екрана виконано самостійно)

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

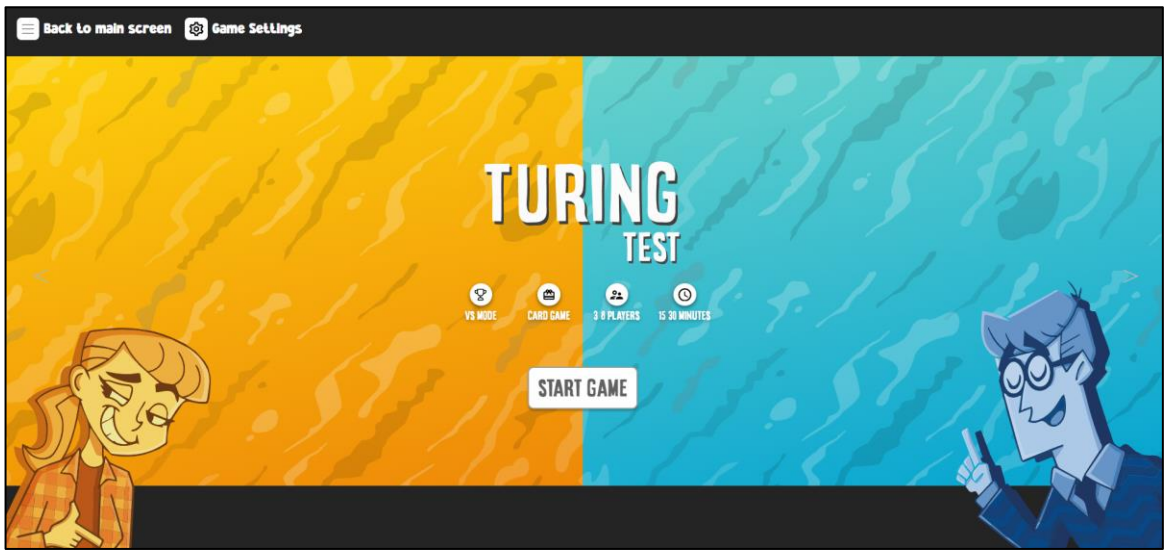


Рисунок 3.5 – Вибір гри Turing Test (знімок екрана виконано самостійно)

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

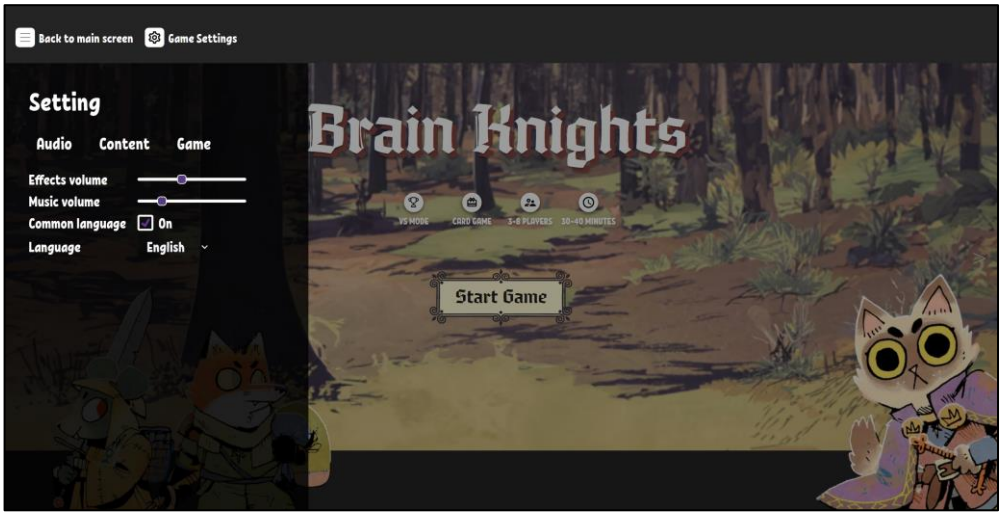


Рисунок 3.6 – Налаштуванні ігрового процесу (знімок екрана виконано самостійно)

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.

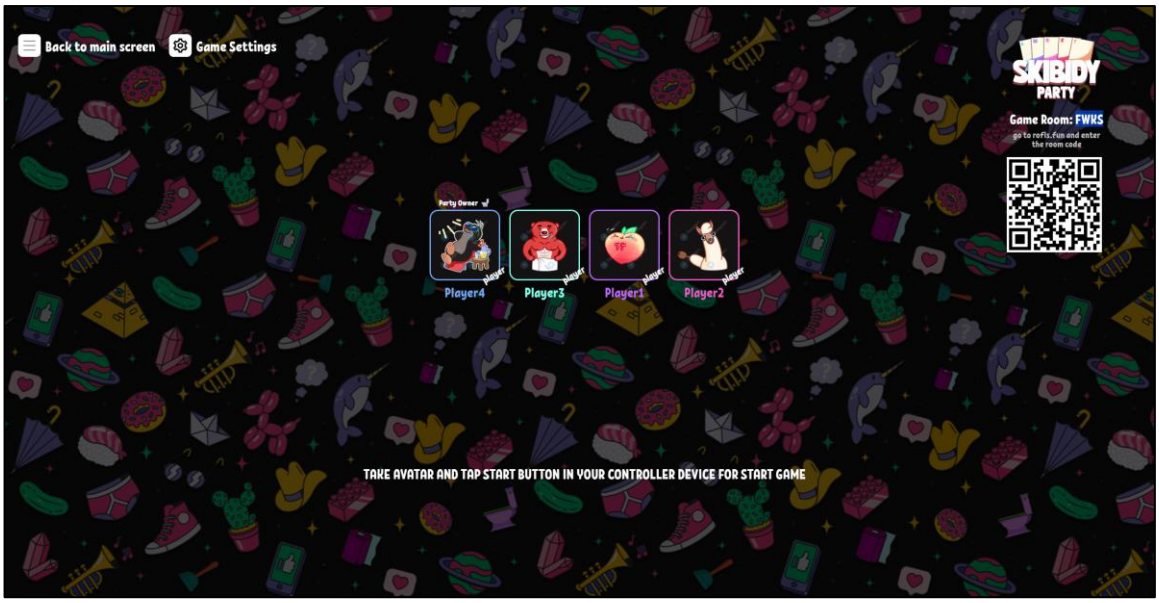


Рисунок 3.7 – Інтерфейс ігрової кімнати в грі Skibidy Party на хості (знімок екрана виконано самостійно)

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

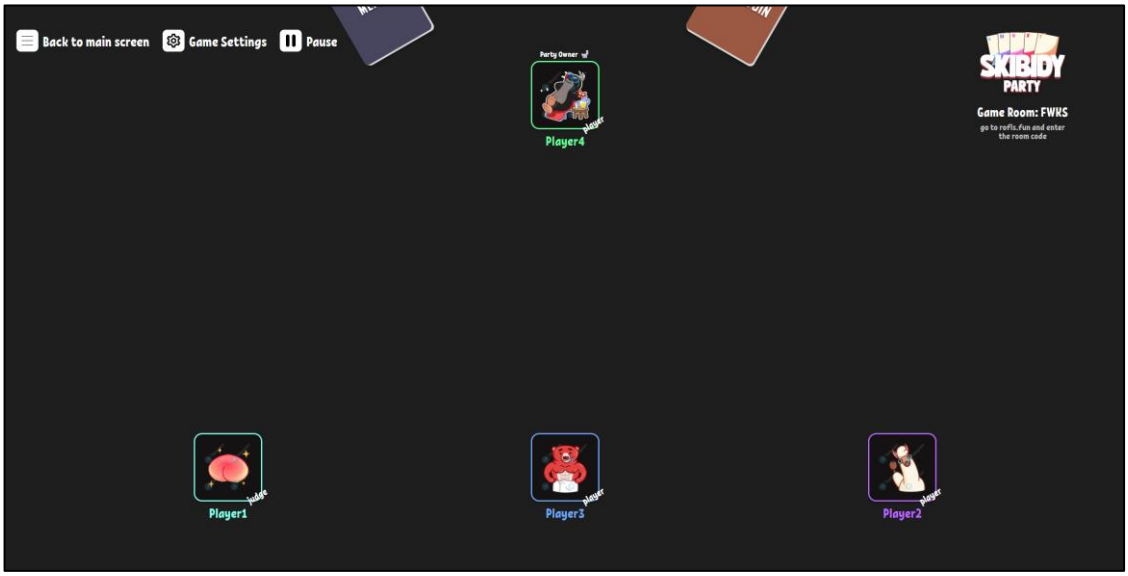


Рисунок 3.8 – Інтерфейс ігрового поля в Skibidy Party на хості (знімок екрана виконано самостійно)

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

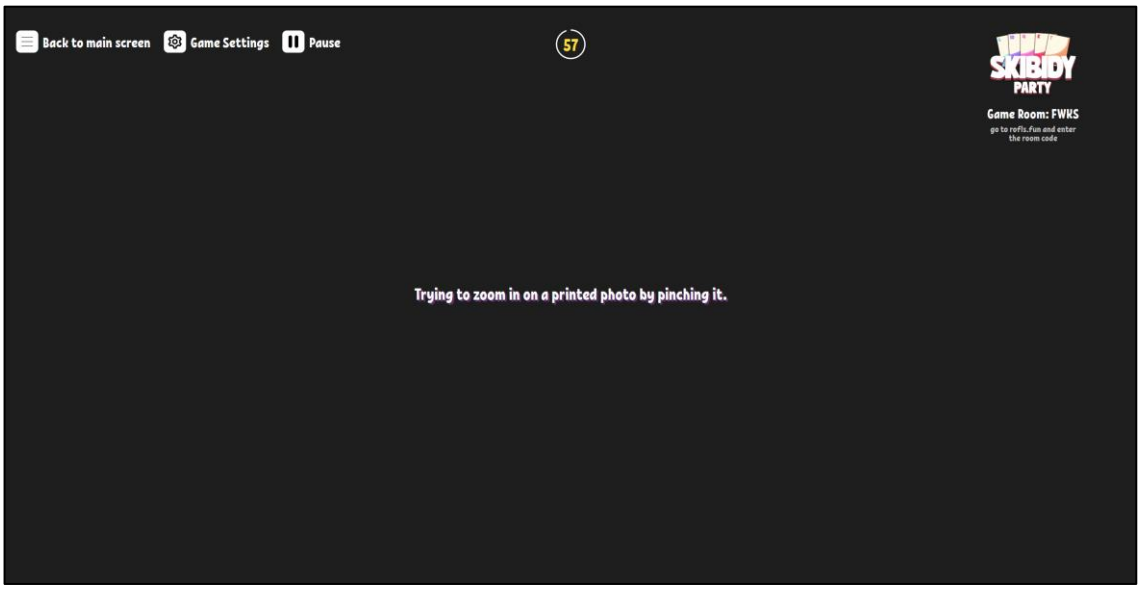


Рисунок 3.9 – Інтерфейс ігрової ситуації в Skibidy Party на хості (знімок екрана виконано самостійно)

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

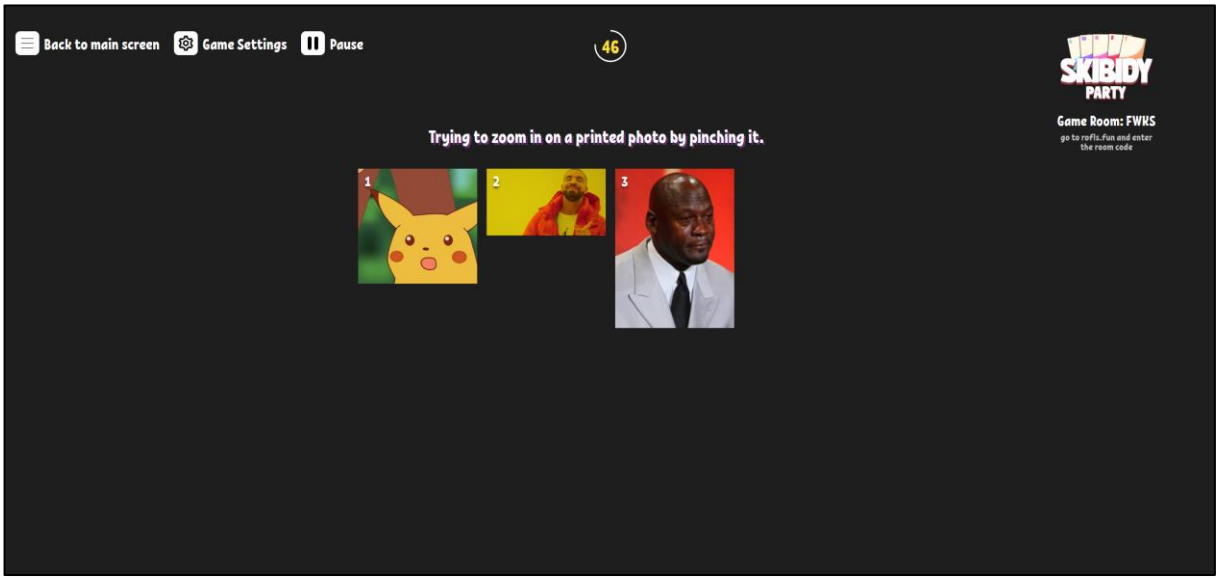


Рисунок 3.10 – Інтерфейс обраних мемів в Skibidy Party на хості (знімок екрана виконано самостійно)

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

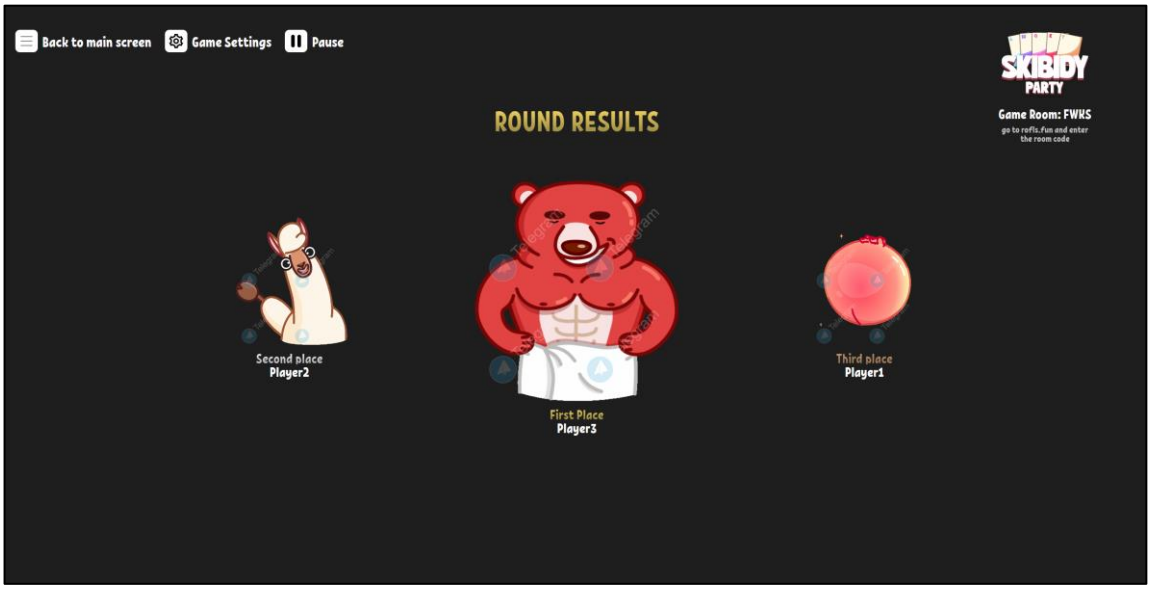


Рисунок 3.11 – Інтерфейс переможців раунду в Skibidy Party на хості (знімок екрана виконано самостійно)

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.

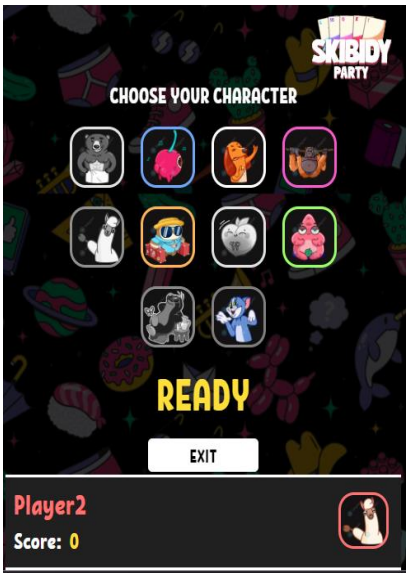


Рисунок 3.13 – Вибір аватару в Skibidy Party на контролері (знімок екрана виконано самостійно)

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

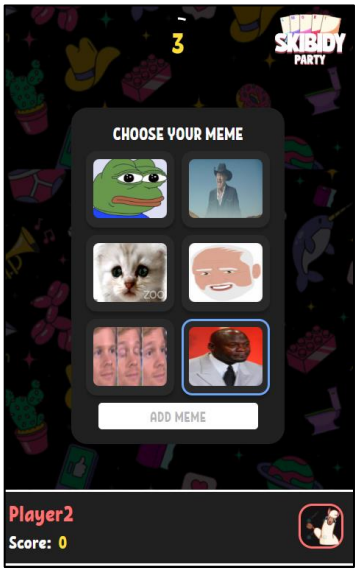


Рисунок 3.14 – Вибір мему в Skibidy Party на контролері (знімок екрана виконано самостійно)

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

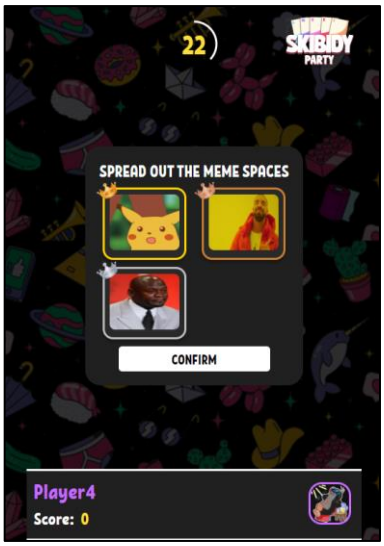


Рисунок 3.15 – Вибір переможного мему в Skibidy Party на контролері (знімок екрана виконано самостійно)

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.

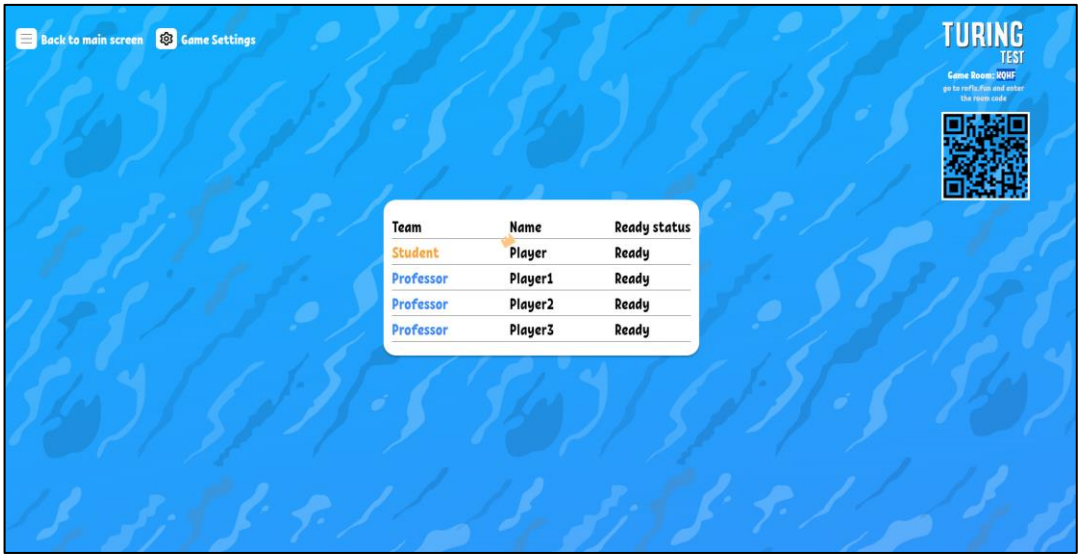


Рисунок 3.16 – Інтерфейс ігрової кімнати в грі Turing Test на хості (знімок екрана виконано самостійно)

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.

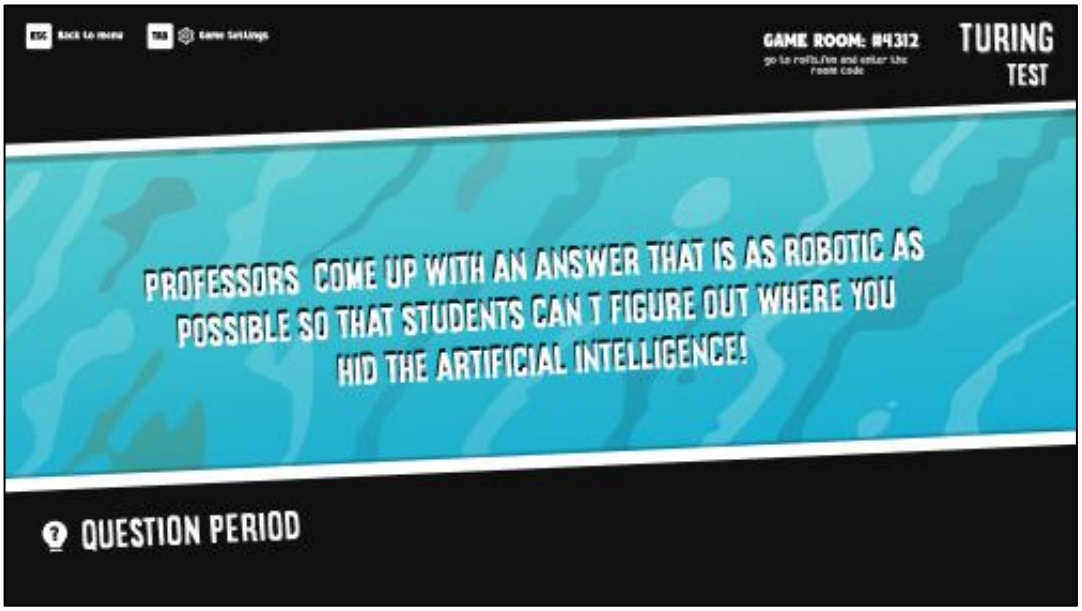


Рисунок 3.18 – Інтерфейс раунду в грі Turing Test на хості (знімок екрана виконано самостійно)

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.

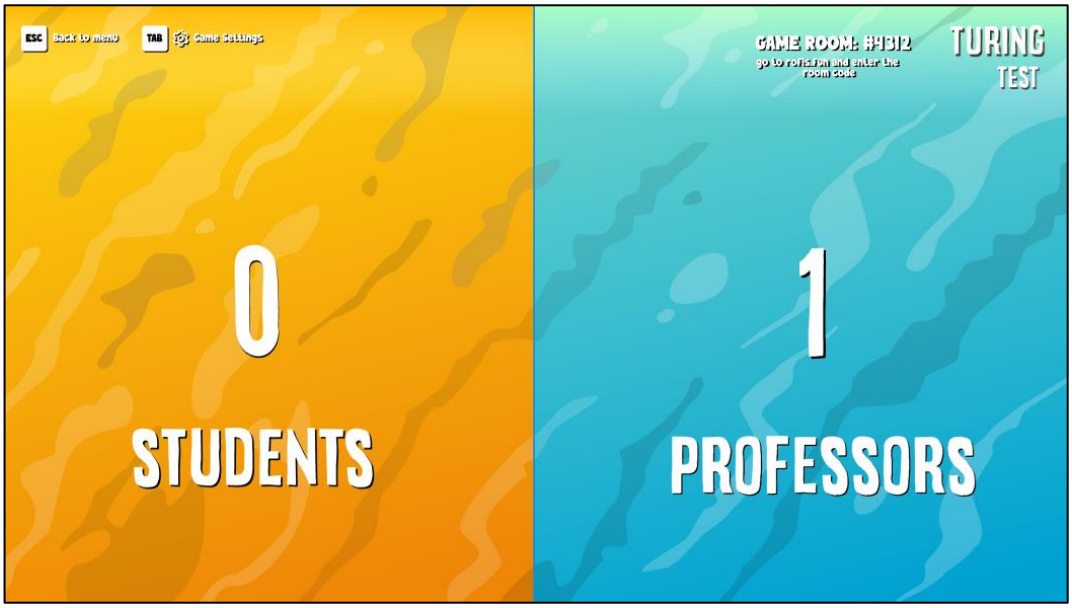


Рисунок 3.23 – Інтерфейс рахунку в грі Turing Test на хості (знімок екрана виконано самостійно)

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.

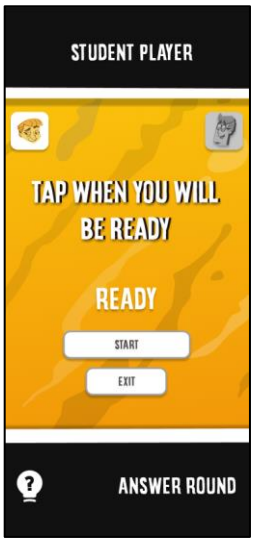


Рисунок 3.24 – Інтерфейс вибору команди в грі Turing Test на котролері (знімок екрана виконано самостійно)

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».

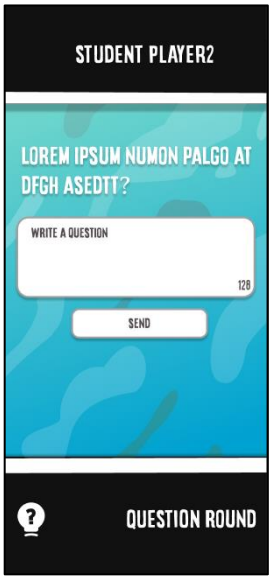


Рисунок 3.26 – Інтерфейс написання відповіді на питання професорами в грі Turing Test на котролері (знімок екрана виконано самостійно)

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.

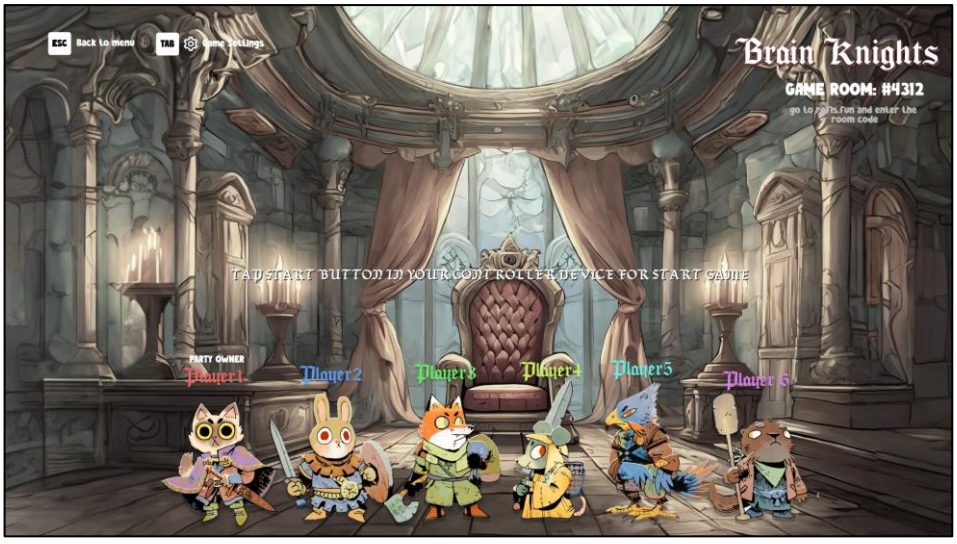


Рисунок 3.29 – Інтерфейс ігрової кімнати на головному екрані гри в грі Brain Knights на хості (знімок екрана виконано самостійно)

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

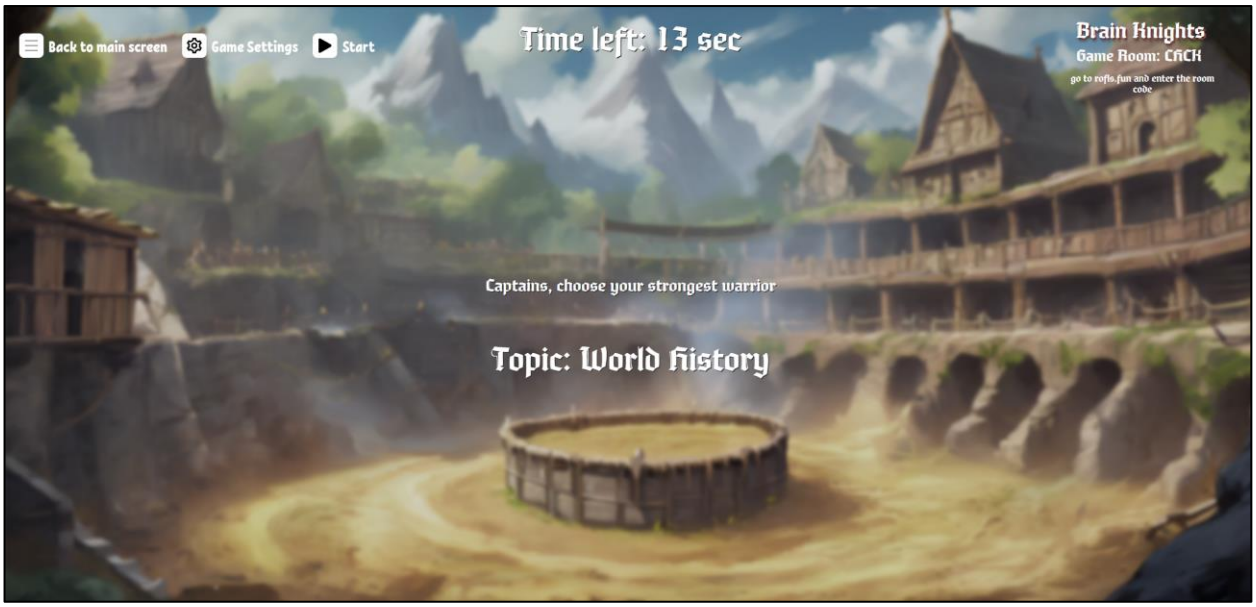


Рисунок 3.30 – Інтерфейс теми раунду в грі Brain Knights на хості (знімок екрана виконано самостійно)

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

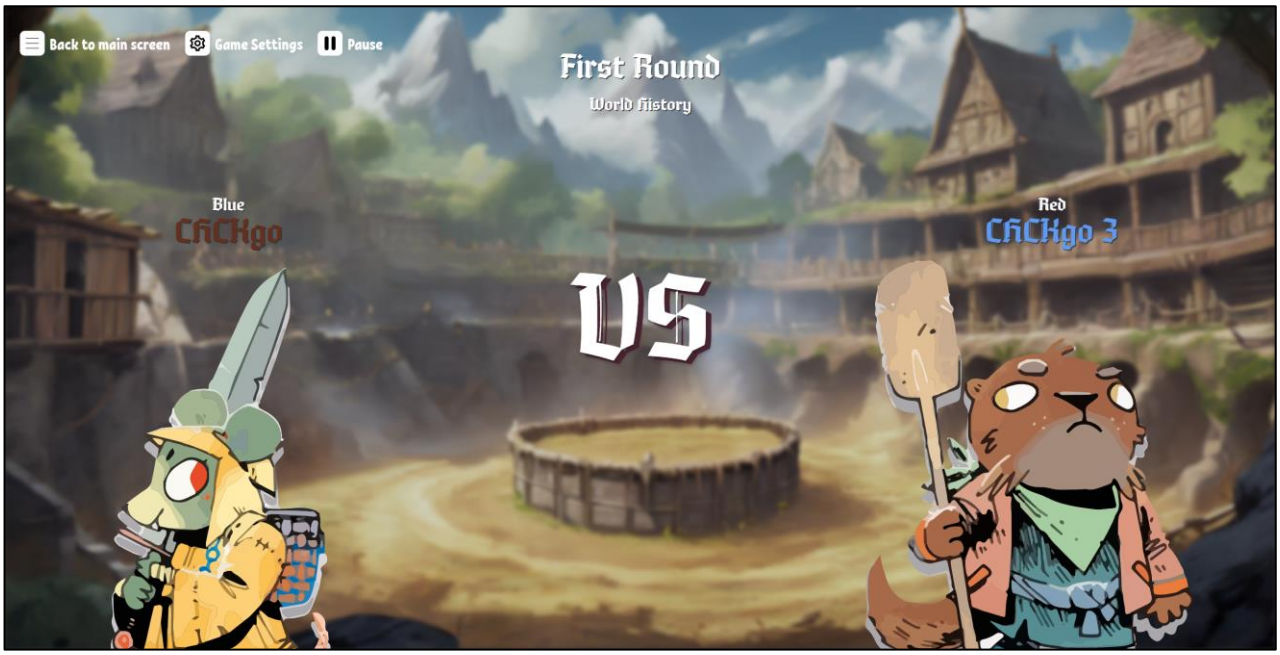


Рисунок 3.31 – Інтерфейс початку бліц-раунду в грі Brain Knights на хості (знімок екрана виконано самостійно)

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

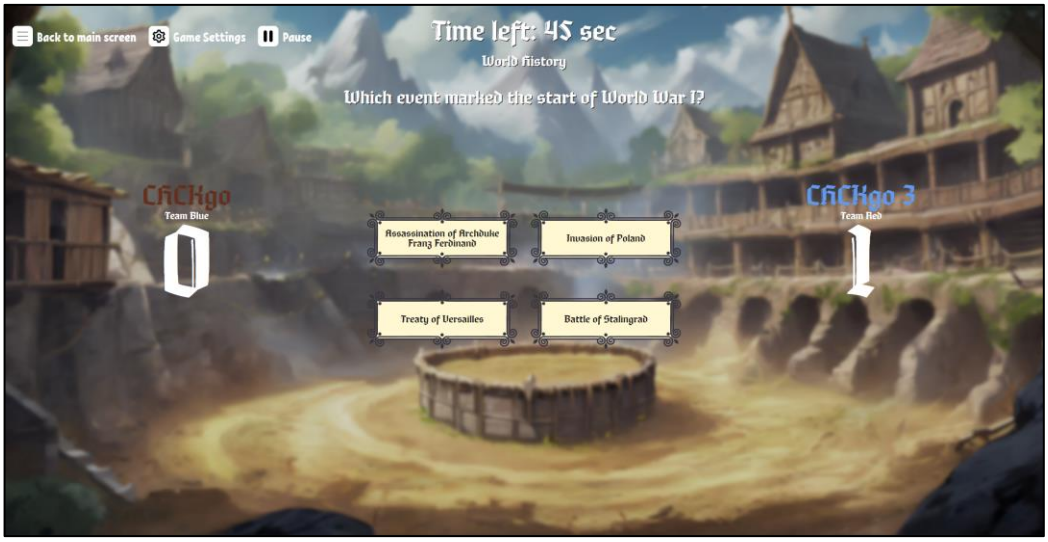


Рисунок 3.32 – Інтерфейс ігрового раунду в грі Brain Knights на хості (знімок екрана виконано самостійно)

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.

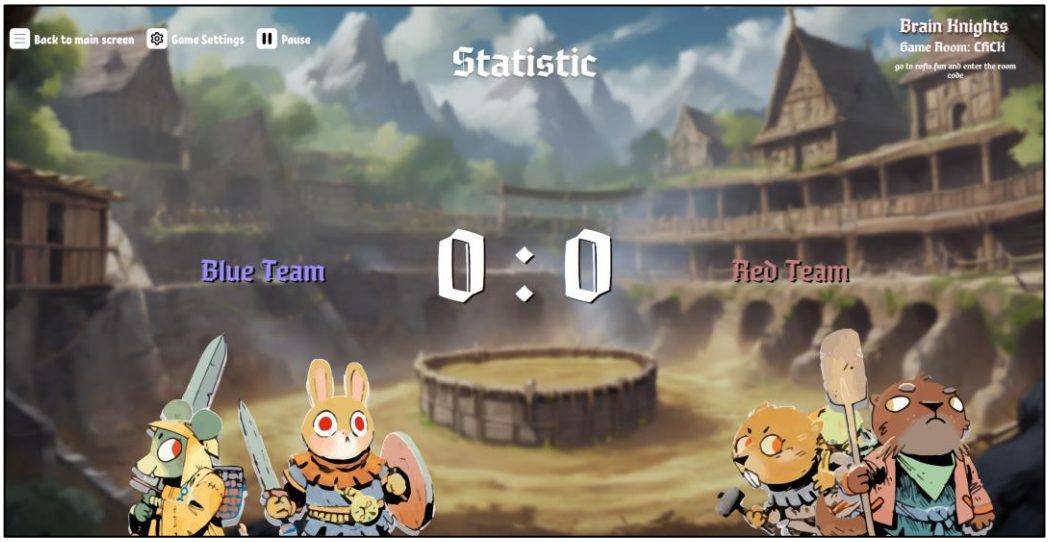


Рисунок 3.34 – Проміжні результати раунду в грі Brain Knights на хості (знімок екрана виконано самостійно)

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.

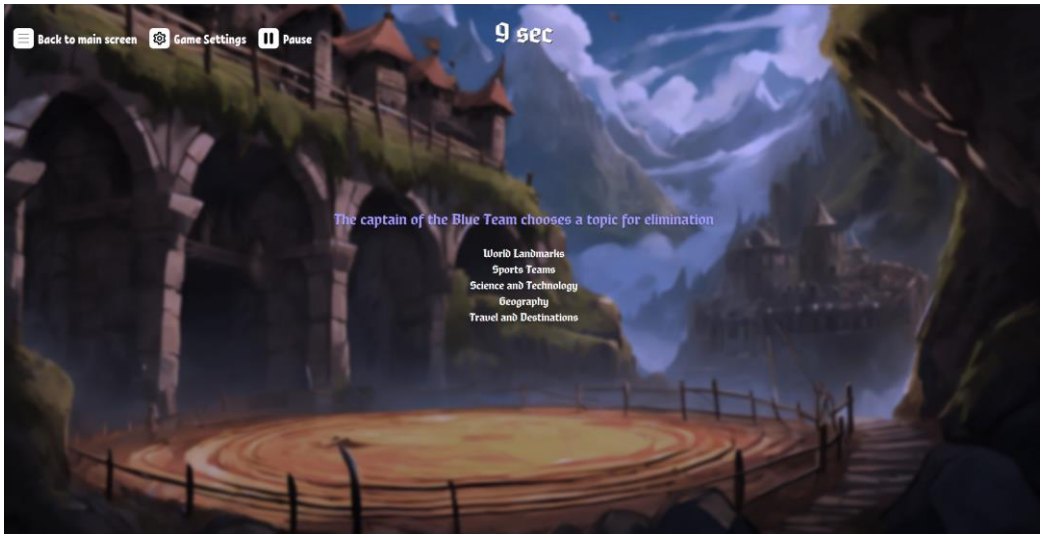


Рисунок 3.35 – Вибір теми для другої стадії в грі Brain Knights на хості (знімок екрана виконано самостійно)

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

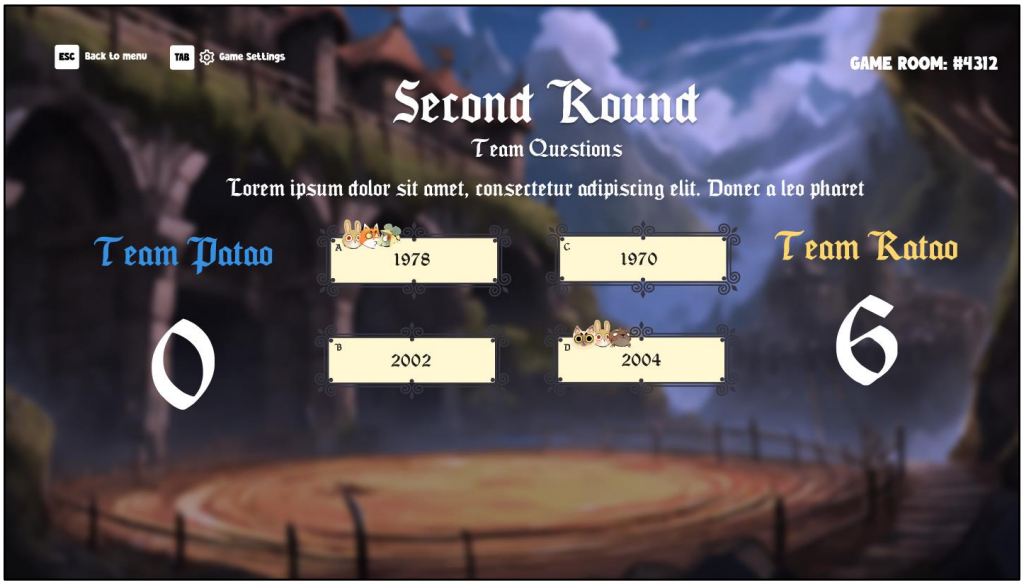


Рисунок 3.36 – Ігровий раунд у другій стадії в грі Brain Knights на хості (знімок екрана виконано самостійно)

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.

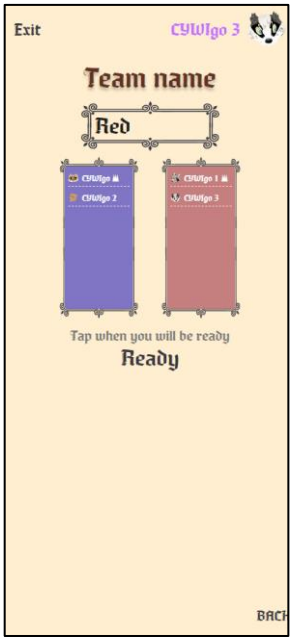


Рисунок 3.38 – Вибір команди в грі Brain Knights на хості (знімок екрана виконано самостійно)

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

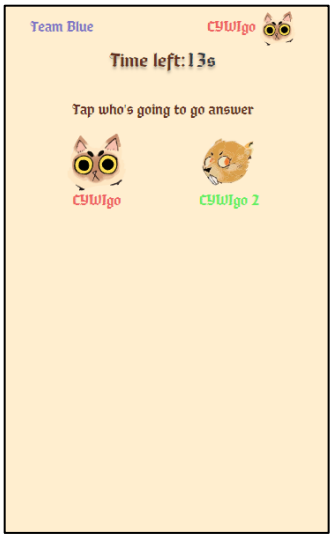


Рисунок 3.39 – Вибір гравця для відповіді в грі Brain Knights на хості (знімок екрана виконано самостійно)

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

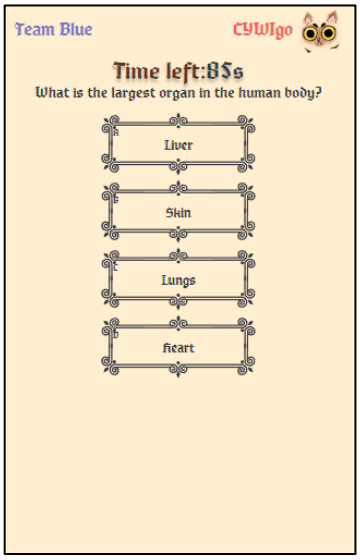


Рисунок 3.40 – Вибір відповіді на запитання в грі Brain Knights на хості (знімок екрана виконано самостійно)

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 слід 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:	Розподіл місць суддею серед найкращих мемів у грі Skibidy Party		
Власник тесту:	Бухало Володимир Олександрович		
Дата створення:	20.05.2024		
Мета тесту:	Перевірити правильність розподілу місць суддею серед найкращих мемів у грі Skibidy Party		
Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	З використанням клієнтської частини створено ігрову сесію гри Skibidy Party	Ігрову сесію створено, і її дані збережено в пам’яті серверного застосунку	Пройдено
2	З використанням клієнтської частини до створеної кімнати підключено чотирьох гравців	Серверний застосунок записав в лог подію підключення кожного гравця, а дані гравців додаються до масиву та зберігаються в пам'яті застосунку	Пройдено

Продовження таблиці Д.1

Передумова			
№	Опис випадку	Очікуваний результат	Висновок
3	З використанням клієнтської частини підтверджено готовність усіх гравців до гри	Серверний застосунок записав в лог подію зміни готовності до гри кожного гравця, а дані готовності гравців були змінені та збережені в пам’яті застосунку	Пройдено
4	Власник кімнати розпочав гру, використовуючи клієнтську частину	Серверний застосунок записав в лог подію початку гри	Пройдено
5	Гравці надіслали свої меми на оцінку, використовуючи клієнтську частину	Серверний застосунок записав в лог подію вибору кожного мема гравцем та зберіг ці дані в пам'яті застосунку	Пройдено
Розподіл місць суддею серед найкращих мемів у грі Skibidy Party			
№	Опис випадку	Очікуваний результат	Висновок
1	На клієнтській частині суддя натиснув вперше на перший із трьох мемів	Серверний застосунок записав в лог подію призначення першого місця для першого мема суддею та зберіг ці дані в пам'яті застосунку	Пройдено
2	На клієнтській частині суддя натиснув вдруге на перший із трьох мемів	Серверний застосунок записав в лог подію призначення другого місця для першого мема суддею та зберіг ці дані в пам'яті застосунку	Пройдено

Продовження таблиці Д.1

Розподіл місць суддею серед найкращих мемів у грі Skibidy Party			
№	Опис випадку	Очікуваний результат	Висновок
3	На клієнтській частині суддя натиснув втретє на перший із трьох мемів	Серверний застосунок записав в лог подію призначення третього місця для першого мема суддею та зберіг ці дані в пам'яті застосунку	Пройдено
4	На клієнтській частині суддя натиснув вчетверте на перший із трьох мемів	Серверний застосунок записав в лог подію призначення першого місця для першого мема суддею та зберіг ці дані в пам'яті застосунку	Пройдено
5	На клієнтській частині суддя натиснув вперше на другий із трьох мемів	Серверний застосунок записав в лог подію призначення другого місця для другого мема суддею та зберіг ці дані в пам'яті застосунку	Пройдено
6	На клієнтській частині суддя натиснув вперше на третій із трьох мемів	Серверний застосунок записав в лог подію призначення третього місця для третього мема суддею та зберіг ці дані в пам'яті застосунку	Пройдено

Кінець таблиці Д.1

Розподіл місць суддею серед найкращих мемів у грі Skibidy Party			
№	Опис випадку	Очікуваний результат	Висновок
7	На клієнтській частині суддя натиснув кнопку для надсилання свого вибору	Серверний застосунок записав в лог подію завершення розподілу місць суддею серед найкращих мемів у грі Skibidy Party та зберіг дані цього розподілу в пам'яті застосунку	Пройдено
Результати тестування			
Тестувальник: Бухало В. О.	Дата прогону тесту: 20.05.2024	Результат тесту (P/F/B): ПРОЙДЕНО (P)	

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

Інформація про тест-кейс			
Ідентифікатор тесту:	Тест-кейс №2		
Опис функції:	Початок ігрової сесії гри Skibidy Party		
Власник тесту:	Бухало Володимир Олександрович		
Дата створення:	20.05.2024		
Мета тесту:	Перевірити можливість розпочати ігрову сесію гри Skibidy Party		
Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	З використанням клієнтської частини створено ігрову сесію гри Skibidy Party	Ігрову сесію створено, і її дані збережено в пам’яті серверного застосунку	Пройдено

Продовження таблиці Д.2

Початок ігрової сесії гри Skibidy Party			
№	Опис випадку	Очікуваний результат	Висновок
1	З використанням клієнтської частини до створеної кімнати підключено першого гравця	Серверний застосунок записав в лог подію підключення гравця, його дані додаються до масиву та зберігаються в пам’яті застосунку	Пройдено
2	З використанням клієнтської частини підтверджено готовність першого гравця	Серверний застосунок записав в лог подію зміни готовності гравця, його дані готовності були змінені та збережені в пам’яті застосунку	Пройдено
3	З використанням клієнтської частини власник ігрової сесії натиснув на кнопку, що розпочинає ігрову сесію	Серверний застосунок порівняв кількість гравців у кімнаті з мінімальною необхідною кількістю, що дорівнює чотирьом, та перевірив підтверджений статус готовності кожного гравця. Ігрова сесія не розпочалася через недостатню кількість гравців	Пройдено
4	З використанням клієнтської частини до створеної кімнати підключено другого гравця	Серверний застосунок записав в лог подію підключення гравця, його дані додаються до масиву та зберігаються в пам’яті застосунку	Пройдено

Продовження таблиці Д.2

Початок ігрової сесії гри Skibidy Party			
№	Опис випадку	Очікуваний результат	Висновок
5	З використанням клієнтської частини підтверджено готовність другого гравця	Серверний застосунок записав в лог подію зміни готовності гравця, його дані готовності були змінені та збережені в пам’яті застосунку	Пройдено
6	З використанням клієнтської частини власник ігрової сесії натиснув на кнопку, що розпочинає ігрову сесію	Серверний застосунок порівняв кількість гравців у кімнаті з мінімальною необхідною кількістю, що дорівнює чотирьом, та перевірив підтверджений статус готовності кожного гравця. Ігрова сесія не розпочалася через недостатню кількість гравців	Пройдено
7	З використанням клієнтської частини до створеної кімнати підключено третього гравця	Серверний застосунок записав в лог подію підключення гравця, його дані додаються до масиву та зберігаються в пам’яті застосунку	Пройдено
8	З використанням клієнтської частини підтверджено готовність третього гравця	Серверний застосунок записав в лог подію зміни готовності гравця, його дані готовності були змінені та збережені в пам’яті застосунку	Пройдено

Продовження таблиці Д.2

Початок ігрової сесії гри Skibidy Party			
№	Опис випадку	Очікуваний результат	Висновок
9	З використанням клієнтської частини власник ігрової сесії натиснув на кнопку, що розпочинає ігрову сесію	Серверний застосунок порівняв кількість гравців у кімнаті з мінімальною необхідною кількістю, що дорівнює чотирьом, та перевірив підтверджений статус готовності кожного гравця. Ігрова сесія не розпочалася через недостатню кількість гравців	Пройдено
10	З використанням клієнтської частини до створеної кімнати підключено четвертого гравця	Серверний застосунок записав в лог подію підключення гравця, його дані додаються до масиву та зберігаються в пам’яті застосунку	Пройдено
11	З використанням клієнтської частини підтверджено готовність четвертого гравця	Серверний застосунок записав в лог подію зміни готовності гравця, його дані готовності були змінені та збережені в пам’яті застосунку	Пройдено

Кінець таблиці Д.2

Початок ігрової сесії гри Skibidy Party			
№	Опис випадку	Очікуваний результат	Висновок
12	З використанням клієнтської частини власник ігрової сесії натиснув на кнопку, що розпочинає ігрову сесію	Серверний застосунок порівняв кількість гравців у кімнаті з мінімальною необхідною кількістю, що дорівнює чотирьом, та перевірив підтверджений статус готовності кожного гравця. Серверний застосунок записав в лог подію початку ігрової сесії та зберіг дані зміни статусу ігрової сесії. Ігрова сесія розпочата	Пройдено
Результати тестування			
Тестувальник: Бухало В. О.	Дата прогону тесту: 20.05.2024	Результат тесту (P/F/B): ПРОЙДЕНО (P)	

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

Інформація про тест-кейс	
Ідентифікатор тесту:	Тест-кейс №3
Опис функції:	Розподіл ролей у грі Turing Test
Власник тесту:	Бухало Володимир Олександрович
Дата створення:	20.05.2024

Продовження таблиці Д.3

Інформація про тест-кейс			
Мета тесту:		Перевірити коректність розподілу ролей у грі Turing Test	
Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	З використанням клієнтської частини створено ігрову сесію гри Skibidy Party	Ігрову сесію створено, і її дані збережено в пам’яті серверного застосунку	Пройдено
Розподіл ролей у грі Turing Test			
№	Опис випадку	Очікуваний результат	Висновок
1	З використанням клієнтської частини до створеної кімнати підключено першого гравця	Серверний застосунок перевіряє кількість гравців у кімнаті та, відповідно до логіки гри, призначає роль студента першому гравцю. У лог записується подія підключення гравця, його дані та роль додаються до масиву і зберігаються на сервері	Пройдено
2	З використанням клієнтської частини до створеної кімнати підключено другого гравця	Серверний застосунок перевіряє кількість гравців у кімнаті та, відповідно до логіки гри, призначає роль професора другому гравцю. У лог записується подія підключення гравця, його дані та роль додаються до масиву і зберігаються на сервері	Пройдено

Продовження таблиці Д.3

Розподіл ролей у грі Turing Test			
№	Опис випадку	Очікуваний результат	Висновок
3	З використанням клієнтської частини до створеної кімнати підключено третього гравця	Серверний застосунок перевіряє кількість гравців у кімнаті та, відповідно до логіки гри, призначає роль професора третьому гравцю. У лог записується подія підключення гравця, його дані та роль додаються до масиву і зберігаються на сервері	Пройдено
4	З використанням клієнтської частини до створеної кімнати підключено четвертого гравця	Серверний застосунок перевіряє кількість гравців у кімнаті та, відповідно до логіки гри, призначає роль професора четвертому гравцю. У лог записується подія підключення гравця, його дані та роль додаються до масиву і зберігаються на сервері	Пройдено
5	З використанням клієнтської частини до створеної кімнати підключено п’ятого гравця	Серверний застосунок перевіряє кількість гравців у кімнаті та, відповідно до логіки гри, призначає роль студента п’ятому гравцю. У лог записується подія підключення гравця, його дані та роль додаються до масиву і зберігаються на сервері	Пройдено

Продовження таблиці Д.3

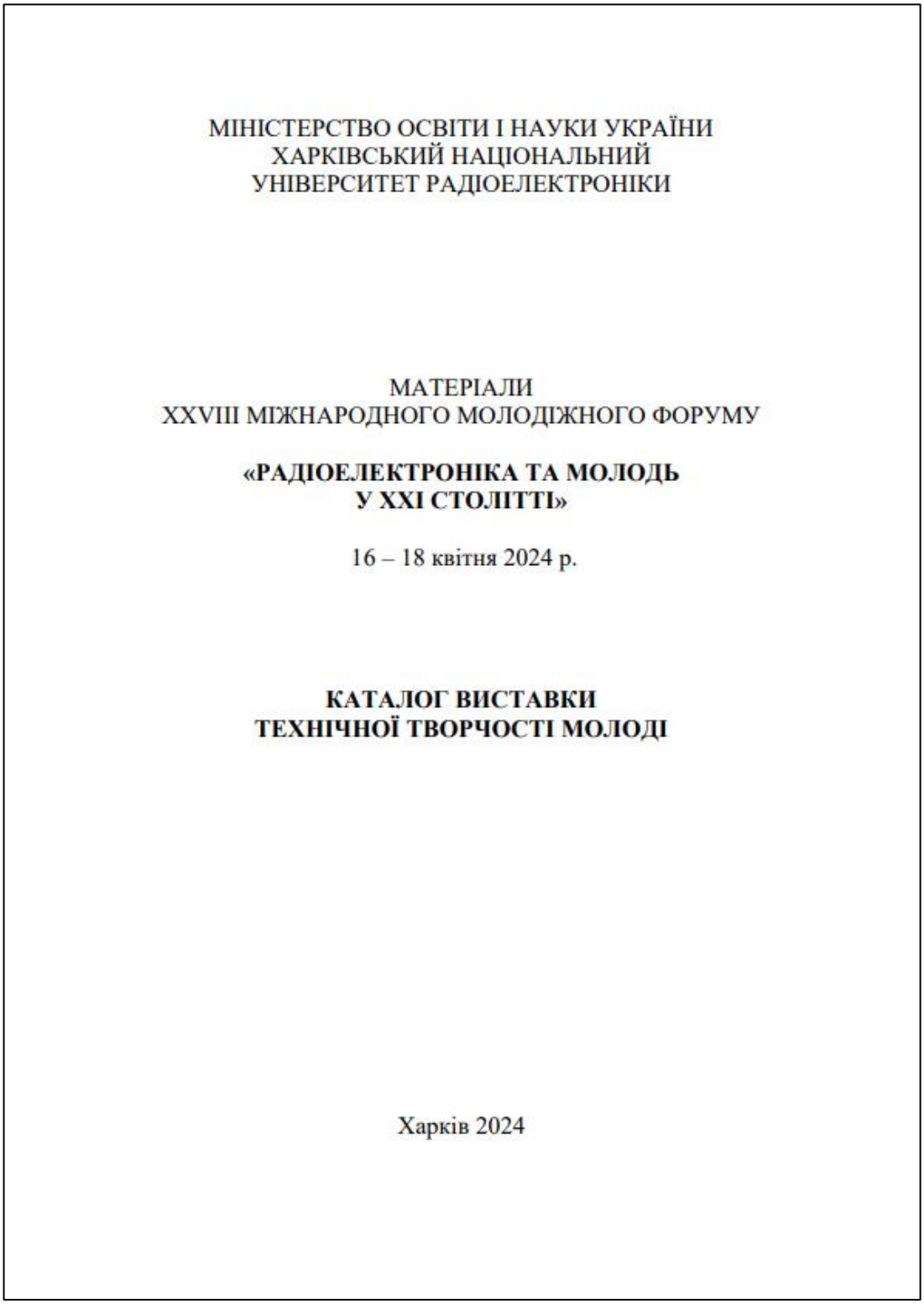
Розподіл ролей у грі Turing Test			
№	Опис випадку	Очікуваний результат	Висновок
6	З використанням клієнтської частини до створеної кімнати підключено шостого гравця	Серверний застосунок перевіряє кількість гравців у кімнаті та, відповідно до логіки гри, призначає роль професора шостому гравцю. У лог записується подія підключення гравця, його дані та роль додаються до масиву і зберігаються на сервері	Пройдено
7	З використанням клієнтської частини до створеної кімнати підключено сьомого гравця	Серверний застосунок перевіряє кількість гравців у кімнаті та, відповідно до логіки гри, призначає роль студента сьомому гравцю. У лог записується подія підключення гравця, його дані та роль додаються до масиву і зберігаються на сервері	Пройдено

Кінець таблиці Д.3

Розподіл ролей у грі Turing Test			
№	Опис випадку	Очікуваний результат	Висновок
8	З використанням клієнтської частини до створеної кімнати підключено восьмого гравця	Серверний застосунок перевіряє кількість гравців у кімнаті та, відповідно до логіки гри, призначає роль професора восьмому гравцю. У лог записується подія підключення гравця, його дані та роль додаються до масиву і зберігаються на сервері	Пройдено
Результати тестування			
Тестувальник: Бухало В. О.	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»

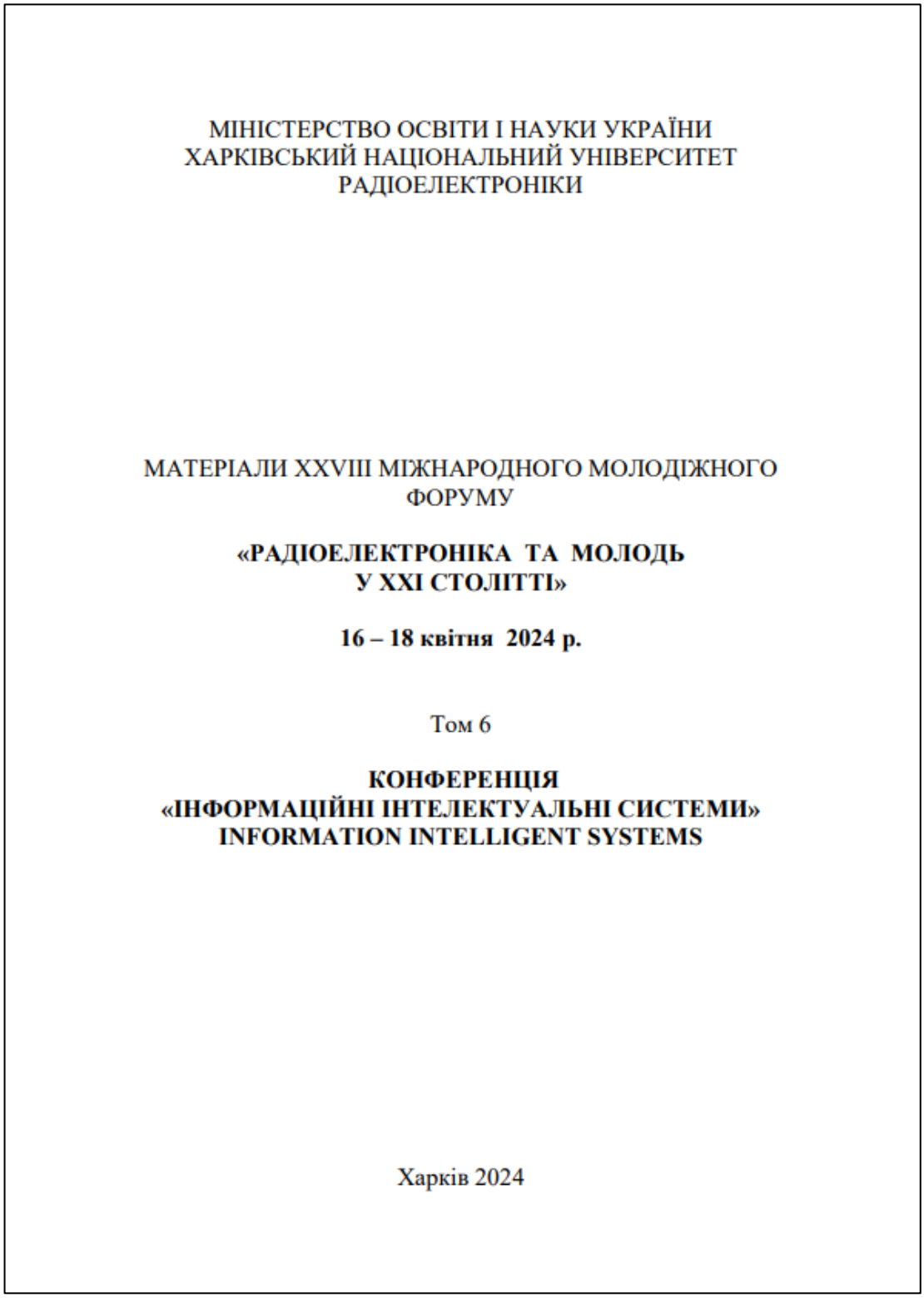
Голова оргкомітету Форуму Ігор РУБАН

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»

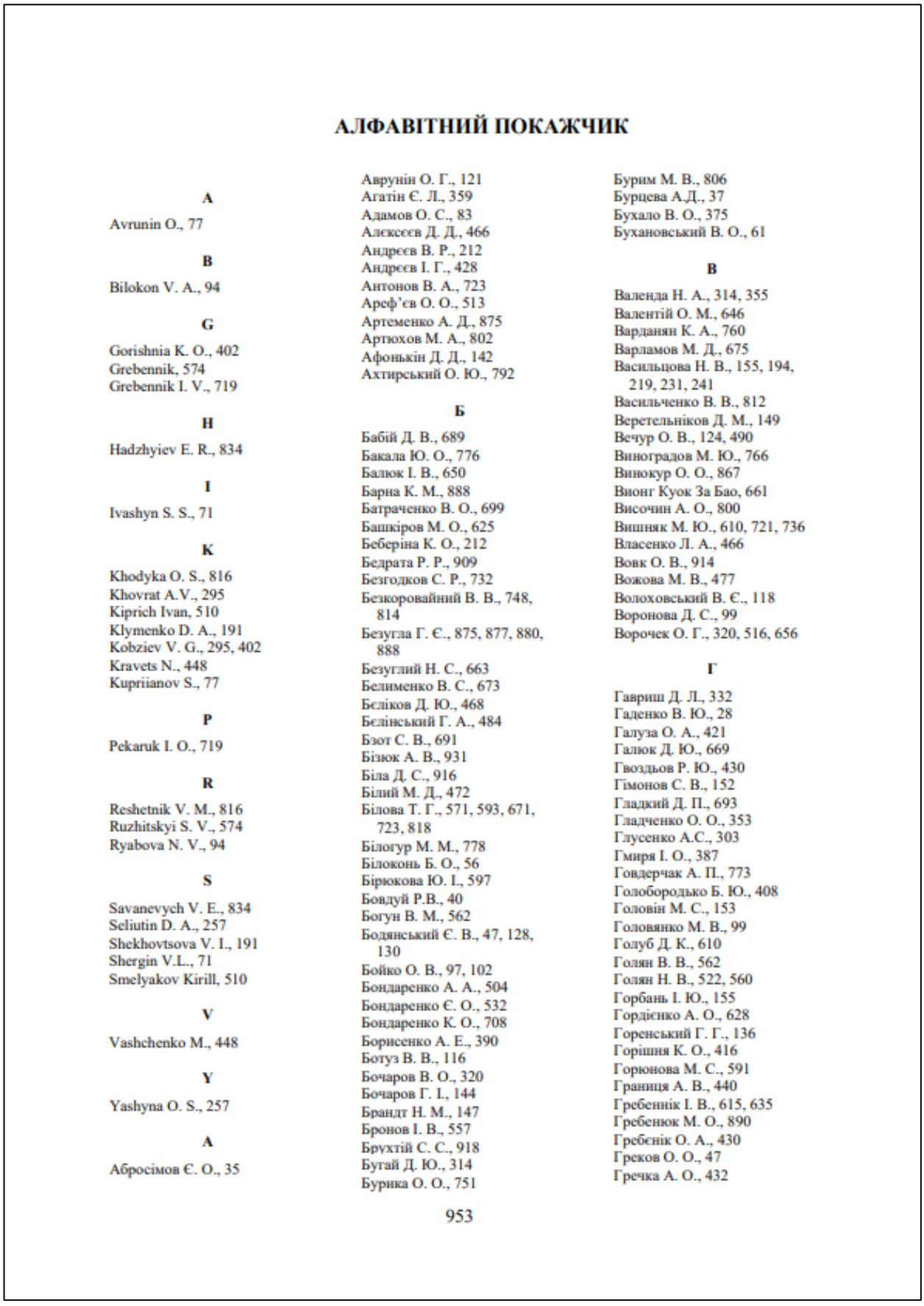
16 – 18 квітня 2024 р.

Том 6

КОНФЕРЕНЦІЯ
«ІНФОРМАЦІЙНІ ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ»
INFORMATION INTELLIGENT SYSTEMS

Харків 2024

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



АЛФАВІТНИЙ ПОКАЖЧИК

953

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

УДК 004.514 DOI: https://doi.org/10.30837/IYF.IIS.2024.384

ВИКОРИСТАННЯ СТРУКТУР ДАНИХ ДЛЯ РЕАЛІЗАЦІЇ ЧЕРГИ ПОДІЙ В ІГРАХ З МЕХАНІКОЮ ОБМЕЖЕННЯ ЧАСУ

Бухало В. О.
Науковий керівник – ст. викл. Новіков Ю. С.
Харківський національний університет радіоелектроніки, каф. ПІ
м. Харків, Україна
e-mail: volodymyr.bukhalo@nure.ua

The current work investigates the use of data structures in developing program code for managing time and events in a gaming environment, where the execution of a particular action is time-constrained. Creating such a tool involves implementing a basic data structure of a queue and modifying it to track and control each player's time, manage the sequence of turns, and respond to events and actions of players within a specified time period. Using the data structure of a queue, the developed software module ensures efficient execution of events in the game process. The work utilized the Node.js runtime environment and the TypeScript programming language.

Ігрова механіка – система або моделювання, заснована на правилах, які полегшують і спонукають користувача досліджувати та вивчати властивості свого простору можливостей за допомогою механізмів зворотного зв'язку [1].

Обмеження часу – це ігрова механіка, яка вимагає від гравця досягнення мети протягом певного періоду часу [2]. Наприклад, ця механіка часто використовується в багатокористувацьких командних іграх в межах однієї кімнати (same-room multiplayer party games), де кожна міні-гра чи подія в ній йде одна за одною і триває протягом певного часу, який регулюється або таймером, або тригером завершення від гравця. Для реалізації таких ігор необхідно мати ефективний механізм для управління чергою подій та часом кожної події окремо.

Дана робота спрямована на реалізацію програмного коду, призначеного для використання в межах ігрового додатку, що забезпечує ефективне керування порядком множини ігрових подій з обмеженим часом їх існування. Для реалізації програмного модулю використовується черга як структура даних для управління ходом гри та змінами стану.

Програмний модуль забезпечує наступні можливості:

– налаштування тривалості виконуваної дії;

– налаштування функцій оберненого виклику;

– створення черги подій;

– зміну виконуваної події у реальному часі при тригерах, ініційованих гравцями;

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

– додавання як окремих, так і групових подій до черги (у разі, якщо елемент черги, який обслуговується, містить декілька подій, то всі вони виконуються асинхронно);

– можливість контролю стану виконуваних подій.

Створення подібного програмного коду досягається завдяки використанню вбудованих інструментів програмної платформи Node.js та мови програмування TypeScript, без використання готових бібліотечних рішень. Налаштування тривалості виконання функцій реалізується за допомогою вбудованої функції «setTimeout», яка приймає у якості аргументу функцію, що має бути виконана, та часовий інтервал, протягом якого вона буде викликана. Ввесь період, що пройшов до фактичного виконання функції, є періодом, у якому вона вже виконується. Для ефективної реалізації черги подій використовується структура даних «черга», яка базується на імплементації масиву.

Реалізовано об'єктно-орієнтований клас, який включає поля для відображення подій, їх черги та поточних виконуваних дій. Після завершення виконання поточних подій, наступний елемент із черги переміщується у поточні події, і їх виконання розпочинається знову. Цей процес триває доти, поки черга не стане порожньою або не буде перервана через дії гравця. Події налаштовані таким чином, що можуть включати в себе різні дії, кожна з яких має свій час виконання та власні функціональні можливості. Використання саме об'єктно-орієнтованої парадигми пояснюється тим, що в розробці ігор широко використовується саме вона, оскільки дозволяє легко моделювати об'єкти гри, їх взаємодію та стан, що сприяє структурованості та розширюваності коду.

Принцип роботи програмного коду графічно наведено на рисунку 1.

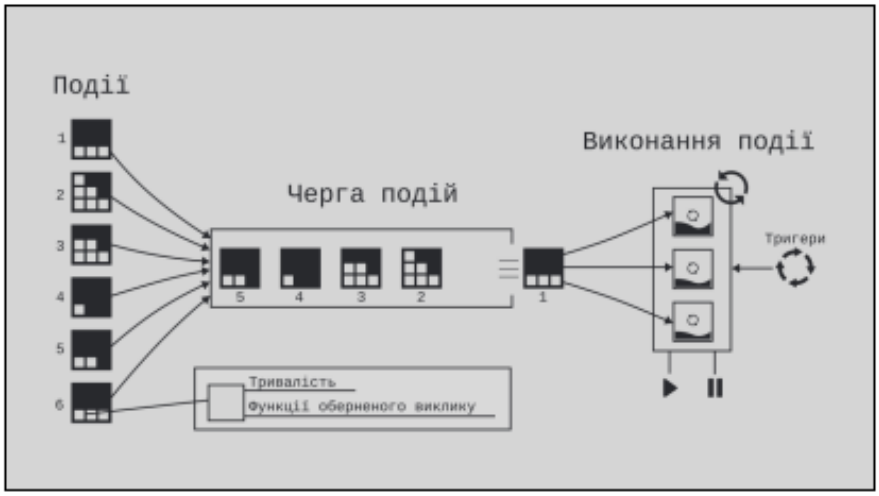


Рисунок 1 – Графічне представлення програмного коду

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

Черга використовується для забезпечення послідовного виконання подій відповідно до заданих часових затримок. Черга діє як структура даних FIFO (перший прийшов, першим пішов) [3], забезпечуючи керування порядком виконання подій. Коли події додаються до черги, вони виходять з неї та обробляються синхронно, даючи гарантію, що дії обробляються в жорстко визначеному порядку, запобігаючи можливі конфлікти або розходження в логіці гри. Однак, логіка подій, які виходять з черги виконується асинхронно, що дозволяє виконувати дії кожної події незалежно від інших, що особливо важливо багатокористувацьких командних іграх в межах однієї кімнати, де гравці можуть взаємодіяти з різними елементами гри одночасно.

Таким чином розроблений програмний код забезпечує ефективне керування часом та подіями у ігровому середовищі з обмеженим часом їх існування. Використання черги дозволяє забезпечити послідовність виконання подій. При цьому розроблений код здатний відстежувати стан подій та реагувати на тригери від гравців, що є важливим аспектом ігрового досвіду.

Список використаних джерел:

1. Часовська А. О. Застосування ігрових механік в комп'ютерних відеоіграх / А. О. Часовська // Радіоелектроніка та молодь у XXI столітті: тези доповідей 27-го Міжнародного молодіжного форуму, 10–12 травня 2023 р. Харків : ХНУРЕ, 2023. Т. 3. С. 155–156.

2. Time limit – TheAlmightyGuru. TheAlmightyGuru.com. URL: http://www.thealmightyguru.com/Wiki/index.php?title=Time_limit#:~:text=A%20time%20limit%20is%20a,a%20way%20to%20add%20tension. (дата звернення: 21.02.2024).

3. What is a FIFO queue and you to implement it efficiently. IME-USP – Instituto de Matemática e Estatística da Universidade de São Paulo. URL: https://www.ime.usp.br/~pf/algorithms/chapters/queues.html (дата звернення: 21.02.2024).

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