ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І. І. МЕЧНІКОВА

(повне найменування закладу вищої освіти)

Факультет математики, фізики та інформаційних технологій

(повне найменування факультету)

Кафедра інформаційних технологій

(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «Магістр»

«Мобільний застосунок для відстеження прогресу у відеоіграх»

(тема кваліфікаційної роботи українською мовою)

«A mobile application for tracking progress in video games»

(тема кваліфікаційної роботи англійською мовою)

Виконав: здобувач денної/заочної форми навчання
спеціальності 122 Комп'ютерні науки
(код, назва спеціальності)
Освітня програма Комп'ютерні науки
(назва)
Притикін Артем Сергійович
(прізвище, ім'я, по-батькові здобувача)

Керівник к.ф.-м.н Ткач Т.Б. _____
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент к.т.н., доцент кафедри інженерії ПЗ національного університету "Одеська політехніка", Зіноватна Світлана Леонідівна
(науковий ступінь, вчене звання, прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
Інформаційних технологій
№ ____ від ____ _____ 2024 р.

Завідувачка кафедри

_____ _____
(підпис) (прізвище,ім'я)

Захищено на засіданні ЕК № _____
протокол № ___ від _____ 2024 р.

Оцінка _____ / _____ / _____
(за національною шкалою/шкалою ECTS/ бали)

Голова ЕК

_____ _____
(підпис) (прізвище,ім'я)

Одеса 2024

АНОТАЦІЯ

Метою магістерської кваліфікаційної роботи є створення інтерактивного веб-додатку для відстеження прогресу користувачів у відеоіграх, що дозволяє гравцям різного рівня зберігати і переглядати свою статистику, вести персональні списки ігор, ділитися досягненнями та посібниками.

Методи розробки базуються на React, CSS-фреймворк Bootstrap для Frontend, Node.js, Express.js, GraphQL для Backend-частини, PostgreSQL для БД, JSON Web Tokens.

Результатом роботи є завершений веб-додаток з функціями геймтрекеру, включаючи синхронізацію досягнень гравців з платформами (Steam, PlayStation, Xbox).

Ключові слова: геймтрекер, відстеження прогресу, ігрова статистика, Інтеграція з платформами Steam, PlayStation, Xbox.

ABSTRACT

The goal of this work is to create an interactive web application for tracking user progress in video games, which allows players of various levels to save and view their statistics, maintain personal lists of games, share achievements and guides.

Development methods are based on React, Bootstrap CSS framework for Frontend, Node.js, Express.js, GraphQL for Backend part, PostgreSQL for DB, JSON Web Tokens.

The result is a complete web application with gametracker features, including synchronization of player achievements with platforms (Steam, PlayStation, Xbox).

Keywords: gametracker, progress tracking, game statistics, Integration with platforms Steam, PlayStation, Xbox.

ЗМІСТ

ВСТУП

З кожним роком кількість геймерів у світі зростає. З психологічної точки зору ігри задовольняють кілька внутрішніх людських потреб. По-перше, вони пропонують відчуття досягнення та винагороди. Коли людина розв’язує головоломку, виграє гонку або проходить рівень, її мозок виділяє дофамін, нейромедіатор, пов’язаний із насолодою та задоволенням. Відеоігри дозволяють гравцям активно взаємодіяти зі світом гри, занурюючись у неї набагато глибше, ніж у фільмах чи книгах. Гравці можуть впливати на хід подій, приймати рішення, які змінюють розвиток сюжету, що створює сильне відчуття залученості.

Відеоігри охоплюють безліч жанрів – від стратегій і рольових ігор до шутерів, спортивних симуляторів і пригодницьких ігор. Це дозволяє кожному знайти щось на свій смак, задовольняючи різні інтереси та уподобання. Багато сучасних відеоігор пропонують багатокористувацькі режими, що відкриває можливість гравцям спілкуватися з друзями або заводити нові знайомства. Ігри створюють спільноти, де люди з різних куточків світу можуть взаємодіяти, співпрацювати або змагатися один з одним.

Багато гравців отримують задоволення від поступового досягнення цілей, поліпшення своїх навичок і підвищення рівнів у грі. Система винагород і досягнень стимулює продовжувати гру та розвиватися в ній. Однією з констант ігрової промисловості є постійний потік нових продуктів і захоплюючих сюжетних ліній. За такої кількості ігор існує потреба у відповіді про те, як грають геймери, які платформи в тренді, що купують геймери і як вони споживають інші форми розваг [1].

Геймтрекери актуальні через зростання кіберспорту, підвищення конкурентоспроможності, складність ігор, а також потребу гравців в аналізі та персоналізованих рекомендаціях. Вони допомагають відстежувати прогрес, покращувати навички, залишатися мотивованими та підтримувати соціальні зв'язки в ігрових спільнотах.

Важко стежити за своєю колекцією відеоігор, особливо якщо вони цифрові та працюють на кількох платформах. Щоб вирішити цю проблему, слід використовувати службу відстеження відеоігор, щоб керувати своєю колекцією, розуміти, що потрібно завершити та що можна купити в майбутньому.

Метою роботи є створення інтерактивного веб-додатку для відстеження прогресу користувачів у відеоіграх, що дозволяє гравцям різного рівня зберігати і переглядати свою статистику, вести персональні списки ігор, ділитися досягненнями та посібниками.

Для досягнення мети слід виконати наступні задачі:

- провести системний аналіз предметної області;
- описати функціональні вимоги до роботи;
- спроектувати архітектуру веб-додатку;
- описати алгоритми роботи його компонентів;
- реалізувати функціонал долатку обраними засобами розробки;
- виконати функціональне тестування.

Об'єктом дослідження виступає система відстеження прогресу та аналізу досягнень у відеоіграх, що включає інтеграцію з ігровими платформами (Steam, PlayStation, Xbox) і соціальні функції для взаємодії гравців.

Предмет дослідження: методи і технології для автоматичного збору і обробки даних про ігровий прогрес і досягнення, алгоритми рекомендацій для гравців, а також підходи до реалізації персоналізованих аналітичних і соціальних функцій у веб-додатку.

1 СИСТЕМНИЙ АНАЛІЗ І КОНЦЕПТУАЛЬНЕ ПРОЕКТУВАННЯ

1.1 Актуальність розробки

Предметна область геймтрекерів охоплює всі аспекти відстеження, аналізу та поліпшення прогресу гравців у відеоіграх. Це багатофункціональні інструменти, що взаємодіють з ігровими платформами, збираючи дані про ігровий процес, щоб надати користувачам корисну інформацію про їх досягнення, статистику та прогрес.

Актуальність геймтрекерів, або додатків для відстеження прогресу в іграх, зростає з кожним роком, і на це є кілька причин:

1. Зростання інтересу до кіберспорту.

Кіберспорт стає все більш популярним, і гравці прагнуть підвищити свої навички, відстежуючи результати ігрових сесій. Геймтрекери дозволяють їм аналізувати свою продуктивність і порівнювати її з іншими гравцями.

2. Підвищення конкурентоспроможності.

У багатьох онлайн-іграх є рейтингові системи, і гравці хочуть підвищити свій рейтинг. Геймтрекери допомагають відслідковувати прогрес, аналізувати помилки та оптимізувати стратегії для досягнення кращих результатів.

3. Соціальний аспект ігор.

Сучасні ігри мають сильний соціальний компонент. Гравці можуть ділитися своїми досягненнями з друзями та спільнотами, обговорювати стратегії та брати участь у спільних заходах. Геймтрекери спрощують цей процес, надаючи зручні інструменти для демонстрації своїх успіхів.

4. Зростаюча складність ігор.

Ігри стають все більш складними, з великою кількістю показників, які потрібно відслідковувати. Геймтрекери допомагають систематизувати цю інформацію та роблять її більш доступною для аналізу.

5. Потреба в персоналізованих рекомендаціях. Геймтрекери часто пропонують поради та рекомендації на основі аналізу ігрових даних. Це

допомагає гравцям покращити свої навички, не витрачаючи багато часу на пошук необхідної інформації самостійно.

6. Підтримка різних ігрових платформ.

Геймтрекери часто підтримують різні платформи та ігри, що дозволяє гравцям зручно відстежувати свій прогрес незалежно від того, де вони грають – на ПК, консолі чи мобільному пристрої.

7. Аналітика для створення контенту.

Стримери та ютубери часто використовують геймтрекери для аналізу своїх ігор і створення контенту. Аналітика допомагає зрозуміти, які моменти варто виділити, та як краще подати матеріал своїй аудиторії.

8. Мотивація та досягнення.

Геймтрекери допомагають гравцям залишатися мотивованими, надаючи чіткі цілі та відстежуючи їх виконання. Це може бути важливим фактором для тих, хто прагне не просто грати для розваги, але й досягати конкретних результатів.

Таким чином, геймтрекери залишаються актуальними через їх здатність покращувати ігровий досвід, допомагати у досягненні цілей та підтримувати соціальні зв'язки між гравцями [2].

Якщо геймер хоче використовувати додаток для відстеження успіхів у пропусунні гри, йому необхідно виконати прив'язку свого облікового запису: при реєстрації або налаштуванні програми користувачу пропонують прив'язати свій ігровий обліковий запис (наприклад, обліковий запис Steam, Riot Games, Blizzard, Epic Games).

Після прив'язування облікового запису програма отримує доступ до ігрових даних користувача, таких як список ігор, в які він граєте, і докладна статистика щодо кожної з них. Ці дані можуть включати: кількість годин, проведених у грі, кількість перемог та поразок, досягнення, рівень, рейтинг та інші метрики.

Геймтрекер збирає та аналізує цю інформацію, надаючи користувачу статистику, графіки, рекомендації та інші корисні дані, щоб він мав

можливість краще зрозуміти свої ігрові звички та успіхи. Ці дані зазвичай автоматично оновлюються після кожного ігрового сеансу, дозволяючи завжди мати актуальну інформацію. Перелік функцій геймтрекерів представлено нижче:

1. Статистика та аналіз гри: програми можуть надавати докладну статистику з кожної гри, включаючи кількість перемог, поразок, співвідношення вбивств і смертей (K/D), середні втрати за матч та багато іншого. Це дозволяє геймерам краще розуміти свої сильні та слабкі сторони.
2. Оновлення в реальному часі: деякі програми дозволяють відстежувати зміни в рейтингу, рівень досвіду або прогрес у виконанні завдань безпосередньо під час гри.
3. Поради та рекомендації: на основі статистики по стилю гри, програми можуть пропонувати рекомендації щодо покращення гри, наприклад, вибір більш відповідної зброї або стратегії.
4. Соціальні функції: багатьом подобається можливість ділитися своїми досягненнями з друзями, порівнювати статистику та змагатися за перше місце у рейтингу.
5. Оновлення та новини: деякі програми надають новини про улюблені ігри, включаючи оновлення, патчі, нові події та турніри [2].

Після входу в систему користувачу запропонують вибрати ігри, які він хоче відстежувати. Для деяких ігор потрібно прив'язати свій ігровий обліковий запис або ввести унікальний ідентифікатор гравця (ID), щоб програма могла отримувати доступ до статистики.

1.2 Аналіз існуючих рішень

Аналіз існуючих рішень у сфері геймтрекерів дозволяє зрозуміти поточний стан ринку, виявити сильні та слабкі сторони популярних додатків, а також визначити можливості для подальшого розвитку.

Backloggd – це місце для віртуального відстеження колекції ігор користувача (рис. 1.1). Backloggd є безкоштовний на всіх платформах. Постійно оновлювання свого журналу, оцінка гри, у які грає користувач, і додавання майбутні ігри у свій список бажань – це маленький перелік можливостей цього ресурсу.

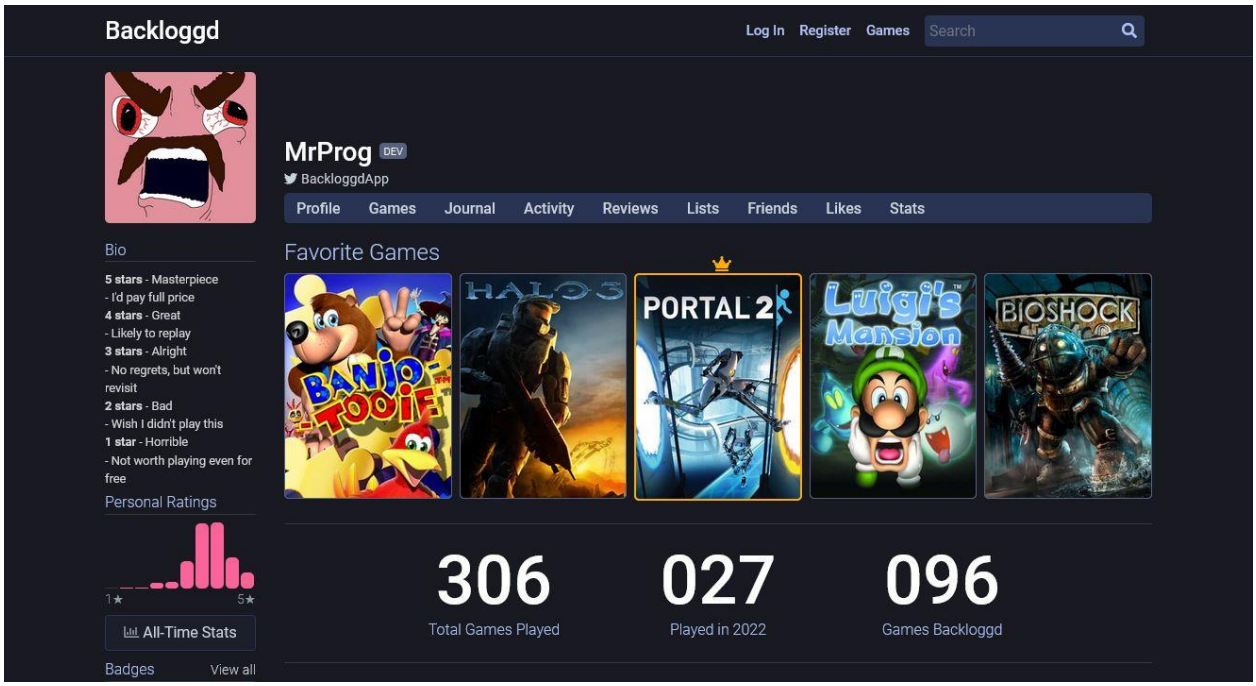

Рисунок 1.1 – Скрін вікна Backloggd

Backloggd дає можливість поділіться своєю ігровою подорожжю з друзями, підписуючись один на одного, щоб бути в курсі останніх ігрових сеансів. У своїй бібліотеці можна легко шукати інформацію за допомогою фільтрів і механізмів сортування, оцінювати ігри та переглядати їх. Крім того, є активна спільнота, яка дозволяє спілкуватися з іншими гравцями та порівнювати свої бібліотеки.

Backloggd надає розширену статистику бібліотеки, як-от найпопулярніші жанри, розвиток бібліотеки за ці роки, улюблені платформи, розподіл прав власності (цифровий, передплатний чи фізичний) тощо. Усе це

розміщено на веб-сайті трекера відеоігор, який добре розроблений, простий у навігації та постійно оновлюється [3].

The Backloggery – це проста служба порівняно з іншими, вона все ще працює як спосіб каталогізувати відеоігри користувача.

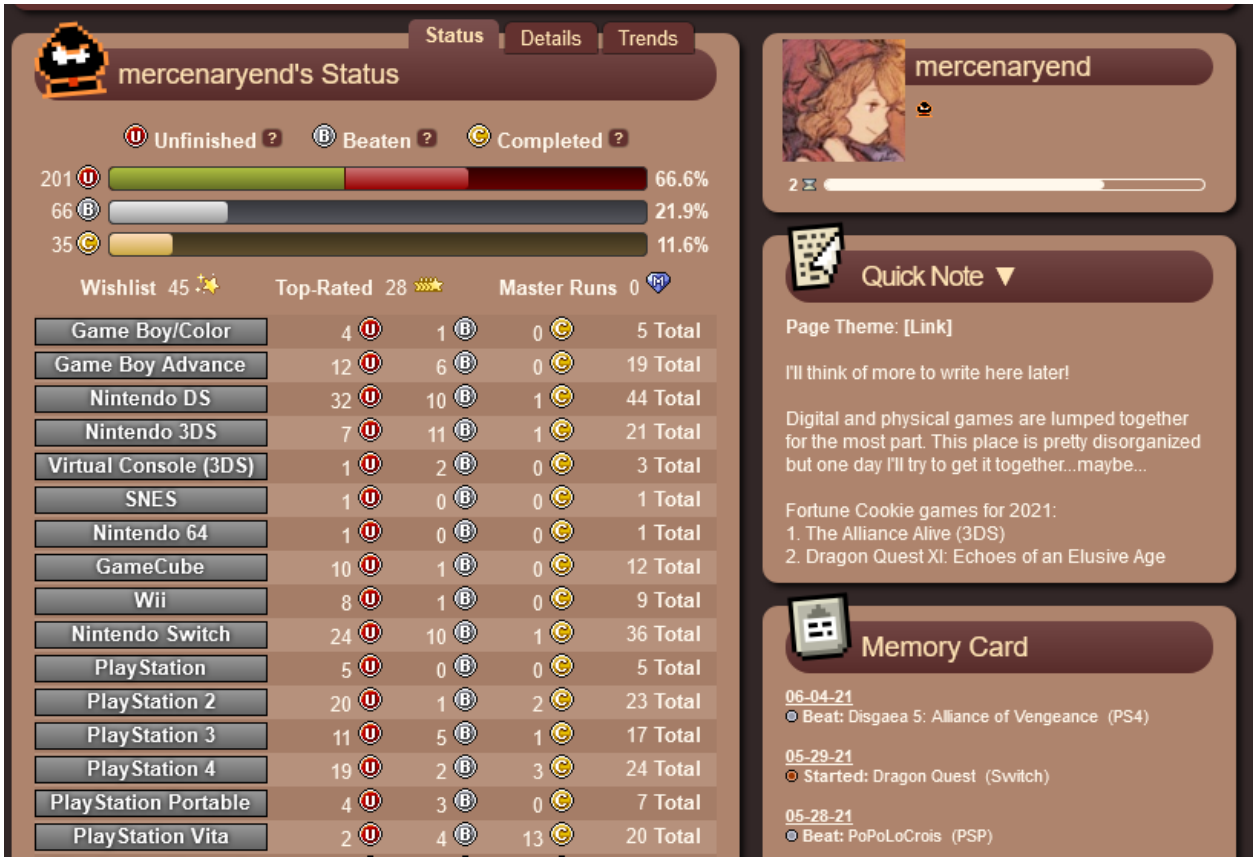

Рисунок 1.2 – Скрін вікна Backloggery

Найбільша відмінність між Backloggery та іншими системами полягає в тому, що вона не забезпечує інтеграцію з базою даних відеоігор. Замість додавання записів про конкретні ігри до облікового запису, користувачу буде запропоновано кілька порожніх полів. Користувач вводить назву, систему та регіон гри разом із статусом її завершення [3].

HowLongToBeat (рис. 1.3) було розроблено як спосіб дізнатися тривалість гри. Це популярний онлайн-сервіс, який надає інформацію про те, скільки часу займає проходження відеоігор. Сайт є зручним інструментом для

геймерів, які хочуть дізнатися, скільки годин потрібно витратити на ту чи іншу гру, залежно від стилю гри та бажання досягнути певних цілей.

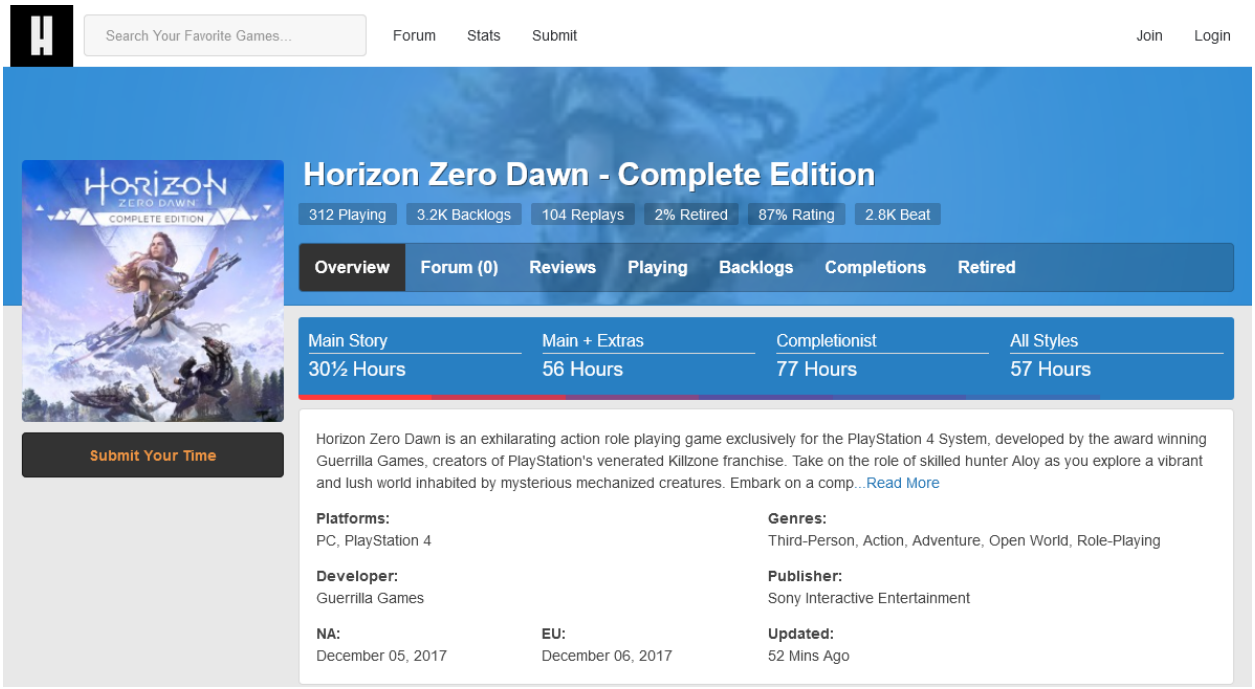

Рисунок 1.3 – Скрін сторінки сайту HowLongToBeat

Основні функції HowLongToBeat:

1. Час проходження ігор: сервіс збирає дані від користувачів, які звітують про те, скільки часу зайняло у них проходження гри.

Час проходження поділяється на кілька категорій:

- Main Story (основна сюжетна лінія): час, необхідний для завершення основного сюжету гри;
- Main+Extra: час, необхідний для завершення основного сюжету разом з додатковими завданнями та контентом;
- Completionist: час, необхідний для повного проходження гри, включаючи всі побічні квести, досягнення і секрети;
- All Styles: середній час, що включає різні стилі проходження.

2. Пошук ігор: користувачі можуть шукати ігри за назвою і переглядати детальну інформацію про кожну гру, включаючи оцінки

користувачів, жанр, платформи та додаткові деталі, такі як дата виходу і видавець.

3. Користувацький внесок: будь-який зареєстрований користувач може внести свої дані щодо часу проходження гри, що допомагає підтримувати актуальність і точність інформації.
4. Персональні списки: користувачі можуть створювати власні списки ігор, що вони хочуть пройти, або відстежувати ігри, які вже завершені.
5. Фільтри і сортування: сервіс дозволяє фільтрувати ігри за різними критеріями, такими як час проходження, жанр або платформа, що спрощує пошук відповідних ігор [4].

Останній аналог займає одну з топових сходинок серед користувачів. GameTrack (рис. 1.4) надає інформацію про розмір ринку та поведінку гравців. Його проводить IPSOS MORI, і це опитування для відстеження в багатьох країнах, покликане отримати ключові дані про те, як люди грають у відеоігри у Великобританії, Франції, Іспанії та Німеччині.

GameTrack включає всі пристрої, які можна використовувати для гри у відеоігри – від ПК і ноутбуків, ігрових консолей і портативних ігрових пристроїв до смартфонів і планшетів, а також нинішніх пристроїв, таких як смарт-телевізори. Так само сюди входять усі формати ігор – від пакетних (нових, уживаних і орендованих) до програм (платних і безкоштовних) до онлайн-ігор (включаючи завантаження, підписки на багатокористувацькі ігри, браузерні ігри та ігри на сайтах соціальних мереж).

Центральними для GameTrack є три ключові показники, кожен з яких розподіляє загальний ринок за окремими пристроями та типами ігор:

1. Обсяг – кількість придбаних ігор (як платних, так і безкоштовних).
2. Цінність - сума грошей, витрачена на ігри.
3. Демографічні показники – хто грає в ігри та як люди грають в ігри.

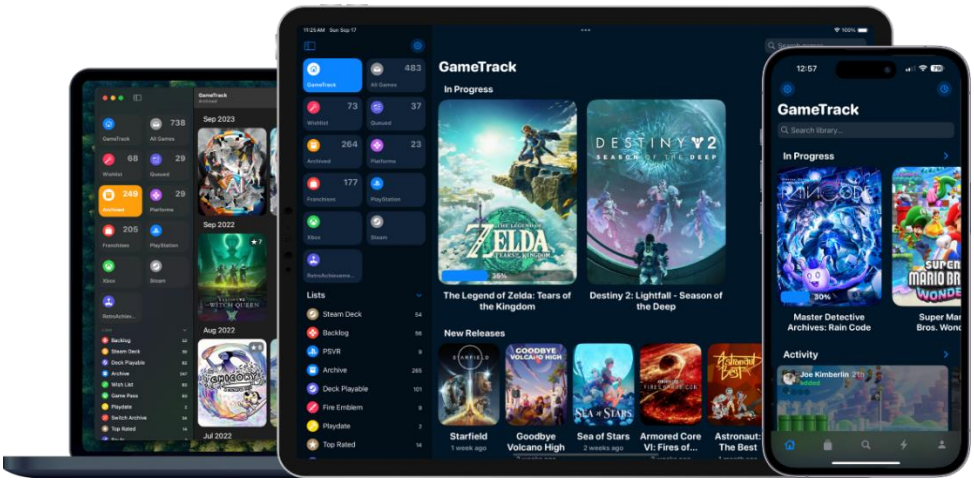

Рисунок 1.4 – Скрін екранів GameTrack

Панель вкладок програми поділена на чотири розділи (рис. 1.5): списки, відкриття, дії та статистика. GameTrack поставляється з трьома списками за замовчуванням: Backlog, Archive і Wish List.

Белог можна далі розділити на «Зараз відтворюється», «Відтворити далі» та «Збірка», а «Архів» — на «Закінчений», «Завершений» і «Залишений». Список побажань упорядковано за датою випуску та може надсилати вам сповіщення, коли перелічені там ігри стануть доступними. [5].

Рисунок 1.5 – Основні розділи GameTrack

Вкладка «Активність» слугує зворотним хронологічним журналом ігор, які користувач додав та до яких списків. Тут також є записи всіх, на кого підписався користувач (кожний контакт включає повну інформацію гравця та посилання на історію спільних чатів). Функція підтримується функцією входу за допомогою Apple і синхронізується між пристроями.

1.3 Опис функціональних та нефункціональних вимог

До функціональних вимог геймтрекеру слід віднести такі функції його роботи, як:

1. Відстеження прогресу: збір і зберігання даних про час, проведений у грі, досягнення, рейтинги.
2. Аналіз даних: автоматичний аналіз ігрової продуктивності та генерація рекомендацій.
3. Інтеграція з платформами: підтримка інтеграції з основними ігровими сервісами (Steam, PlayStation, Xbox).
4. Соціальні функції: можливість ділитися досягненнями, порівнювати статистику з іншими гравцями.
5. Персоналізація: надання рекомендацій на основі стилю гри кожного користувача.

Нефункціональні вимоги:

1. Безпека: захист персональних даних користувачів і безпека з’єднань з ігровими платформами.
2. Масштабованість: система повинна підтримувати одночасне використання великою кількістю користувачів.
3. Продуктивність: швидкий аналіз даних і відображення результатів в інтерфейсі користувача.

1.4 Моделювання бізнес-процесів

IDEF0 – це метод моделювання бізнес-процесів, який дозволяє наочно зобразити структуру системи, її функції та взаємозв'язки між ними. Для геймтрекеру IDEF0 діаграма допоможе визначити основні функції системи, їхні входи та виходи, механізми виконання, а також контрольні елементи.

На рисунку 1.6 представлено діаграму IDEF0 для основної функції геймтекеру «Відстежження та аналіз прогресу в іграх».

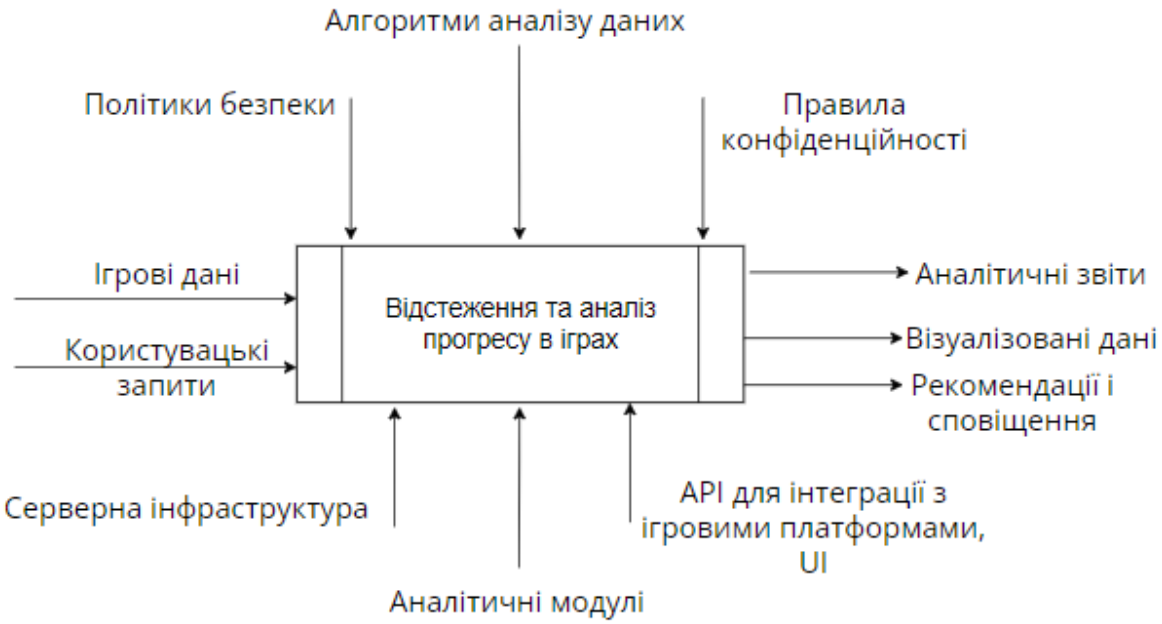

Рисунок 1.6 – Представлення основної функції геймтрекеру у контексті IDEF0

1. Вхід (Input):
   - Ігрові дані: статистика матчів, досягнення, рейтинги, час гри, інформація про виконані квести;
   - Користувацькі запити: запити від користувачів на отримання статистики, аналітики або рекомендацій.

2. Вихід (Output):

- Аналітичні звіти: результати аналізу ігрових даних, які включають виявлені тренди, сильні та слабкі сторони гравця;
- Рекомендації: персоналізовані поради щодо покращення гри, оптимізації стратегії тощо;
- Візуалізовані дані: графіки, діаграми, таблиці, які відображають прогрес гравця у зрозумілому вигляді;
- Сповіщення: інформаційні повідомлення, які інформують користувачів про зміни в статистиці або нові рекомендації.

3. Механізми (Mechanisms):

- Серверна інфраструктура: сервери для зберігання і обробки даних, бази даних для зберігання ігрової статистики;
- Аналітичні модулі: програмні компоненти, які відповідають за аналіз зібраних даних і генерацію рекомендацій;
- API для інтеграції з ігровими платформами: інтерфейси, які забезпечують отримання даних від різних ігрових платформ (наприклад, Steam, PlayStation, Xbox);
- Інтерфейс користувача (UI): компоненти, що забезпечують взаємодію користувача з системою, дозволяючи переглядати статистику, звіти та отримувати рекомендації.

4. Контроль (Controls):

- Політики безпеки: набір правил, що забезпечує захист даних користувачів і запобігає несанкціонованому доступу;
- Правила конфіденційності: політики, які визначають, як обробляються та зберігаються персональні дані користувачів, забезпечуючи їх конфіденційність;
- Алгоритми аналізу даних: налаштовані бізнес-правила і алгоритми, які забезпечують правильність і точність аналізу ігрових даних;

- Режими доступу: правила та обмеження щодо того, хто і як може отримувати доступ до різних рівнів інформації і функціональності системи.

Наступний крок у побудові діаграми бізнес-процесів – це декомпозиція основної функції. «Відстеження та аналіз прогресу в іграх» було поділено на 5 скоадових, а саме:

- A1: Збір та обробка ігрових даних;
- A2: Аналіз ігрових даних;
- A3: Генерація рекомендацій;
- A4: Візуалізація даних;
- A5: Управління користувацькими акаунтами.

На рисунку 1.7 представлено діаграму IDEF0 першого рівня.

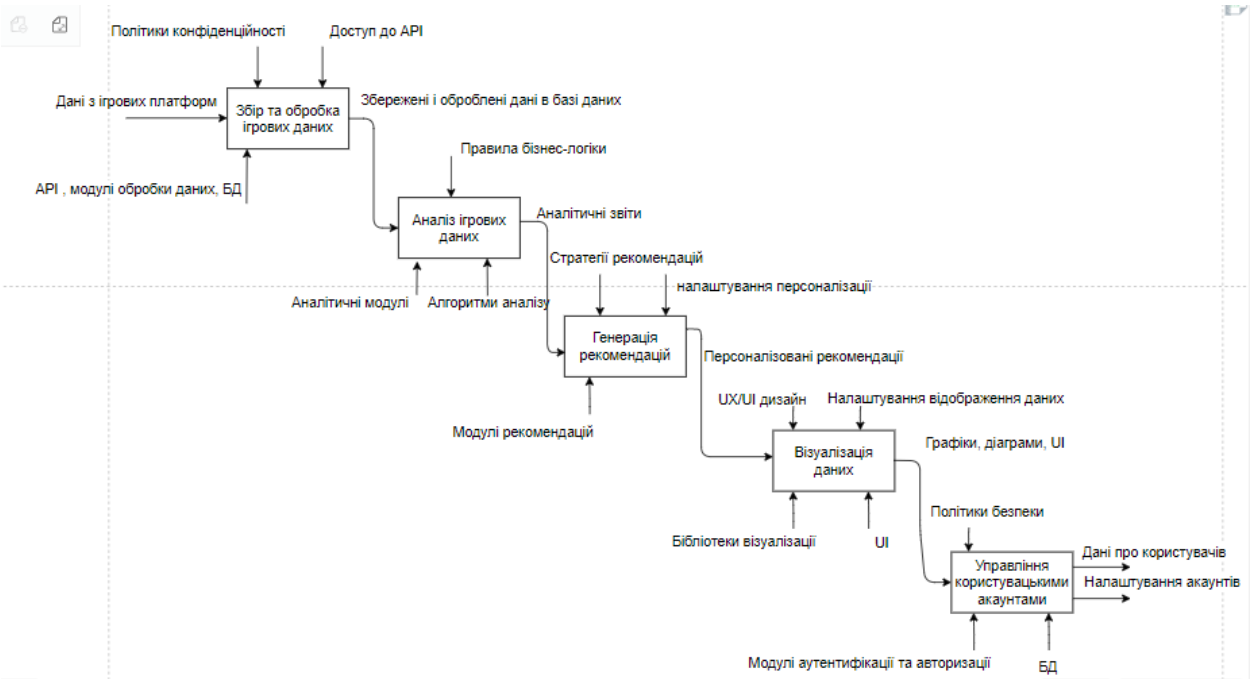

Рисунок 1.7 – Декомпозиція першого рівня основної функції

A1 «Збір та обробка ігрових даних»:

- Входи: дані з ігрових платформ (Steam, PlayStation, Xbox).
- Виходи: збережені і оброблені дані в базі даних, готові до аналізу.

- Механізми: API для інтеграції з ігровими платформами, модулі обробки даних, база даних.
- Контроль: політики конфіденційності, доступ до API.

A2 «Аналіз ігрових даних»:

- Входи: оброблені дані про ігровий прогрес.
- Виходи: аналітичні звіти, виявлені тренди, сильні та слабкі сторони гравця.
- Механізми: аналітичні модулі, алгоритми аналізу.
- Контроль: налаштовані алгоритми аналізу, правила бізнес-логіки.

A3 «Генерація рекомендацій»:

- Входи: Результати аналізу ігрових даних.
- Виходи: Персоналізовані рекомендації для покращення гри.
- Механізми: Модулі рекомендацій, машинне навчання (якщо застосовно).
- Контроль: Стратегії рекомендацій, налаштування персоналізації.

A4 «Візуалізація даних»:

- Входи: аналітичні звіти, збережені дані про прогрес.
- Виходи: візуалізовані графіки, діаграми, інтерфейс користувача.
- Механізми: інтерфейс користувача, бібліотеки візуалізації (наприклад, D3.js).
- Контроль: UX/UI дизайн, налаштування відображення даних.

A5 «Управління користувацькими акаунтами»:

- Входи: дані про користувачів, налаштування акаунтів.
- Виходи: управління доступом, захищені дані акаунтів.
- Механізми: модулі аутентифікації та авторизації, база даних користувачів.
- Контроль: політики безпеки, правила конфіденційності.

Взаємодія між функціями забезпечується за допомогою передачі даних від одного модуля до іншого (наприклад, дані, зібрані на етапі A1, передаються для аналізу на етап A2). Контрольні елементи забезпечують правильність і

безпеку виконання кожної функції, наприклад, політики безпеки контролюють управління користувацькими акаунтами і доступ до API.

IDEF0 для геймтрекеру відображає загальну структуру системи, показує, як різні компоненти взаємодіють між собою, і які вхідні та вихідні дані використовуються в процесі виконання функцій. Така діаграма є корисною для планування архітектури системи та забезпечення її ефективного функціонування.

2 ПРОЕКТУВАННЯ ВЕБ-ДОДАТКУ

2.1 Опис алгоритмів роботи модулів геймтрекеру

Функції геймтрекера спрямовані на полегшення управління ігровим процесом, відстеження прогресу і отримання аналітичної інформації. Нижче наведено пис кожної функції з описом варіанту їх реалізування.

2.1.1 Відмітка відсотків проходження гри і часу

Принцип роботи функції наступний:

- користувач починає гру і відкриває сесію в додатку. додаток запускає секундомір, який рахує проведений час;
- після завершення сесії користувач вимикає секундомір і вручну вводить відсоток проходження гри, який відображає прогрес, досягнутий під час цієї сесії (на основі внутрішніх даних гри або арі платформи);
- додаток використовує дані сесій для прогнозування часу, необхідного для завершення гри, виходячи з середнього темпу проходження.

Приклад розрахунку часу: якщо користувач проходить 25% гри за одну годину і 50% за другу годину, додаток розраховує середній темп проходження і прогнозує, що для завершення гри залишилося 2 години. Формула виглядає так:

Середній відсоток за годину = (25% + 50%) / 2 = 37.5% за годину.

Залишковий відсоток = 100% - 50% = 50%.

Час для завершення гри = 50% / 37.5% = ~1.33 години (приблизно 1 година 20 хвилин).

Відображення в додатку буде наступним:

- користувач бачить час, проведений у грі, та відсоток проходження в інтуїтивно зрозумілому інтерфейсі;

- додаток надає прогноз часу, через який користувач зможе завершити гру [6].

Доступ до Steam API можна отримати через URL: http://api.steampowered.com/ISteamUserStats/GetPlayerAchievements/v0001/(що належить офіційному API Steam). Цей ендпоінт використовується для отримання інформації про досягнення користувачів у конкретних іграх. Тут йде чітка ідентифікація користувача та гри:

- steamid: унікальний ідентифікатор користувача steam, він забезпечує доступ до персональних даних про досягнення;
- appid: ідентифікатор гри, який гарантує, що запит стосується конкретної гри.

Указання «v0001» означає, що використовується перша версія ендпоінта, яка є стабільною та підтримуваною. API-ключ забезпечує ідентифікацію програми, що виконує запит, та контролює доступ до API. Він захищає дані від несанкціонованого використання. Ендпоінт дозволяє отримати: загальну інформацію про користувача, назву гри, список досягнень із їх статусами (отримано/не отримано), час отримання досягнень. Адреса API є легкою для використання в запитах, відповіді мають стандартний формат JSON, що спрощує обробку даних у додатку.

Таким чином, обрана адреса максимально відповідає цілям інтеграції: дозволяє зібрати необхідну інформацію про досягнення користувача в обраній грі з мінімальною складністю.

Після запиту, де:

- appid: 730 – це ID гри Counter-Strike: Global Offensive (CS:GO);
- steamid: 76561198000000000 – приклад Steam ID користувача

було отримано таку інформацію:

```
http://api.steampowered.com/ISteamUserStats/GetPlayerAchievements/v0001/?appid=730&key=XX&steamid=76561198000000000
```

```
{
  "playerstats": {
    "steamID": "76561198000000000",
    "gameName": "Counter-Strike: Global Offensive",
    "achievements": [
      {
        "apiname": "Win_10_Competitive_Matches",
        "achieved": 1,
        "unlocktime": 1625240400
      },
      {
        "apiname": "Get_30_Bomb_Kills",
        "achieved": 0,
        "unlocktime": 0
      },
      {
        "apiname": "Defuse_10_Bombs",
        "achieved": 1,
        "unlocktime": 1625154000
      }
    ]
  }
}
```


Розшифрування параметрів відповіді:

1) steamID: 76561198000000000 – ID користувача;
2) gameName: "Counter-Strike: Global Offensive" – назва гри;
3) achievements:
   - apiname: "Win_10_Competitive_Matches" – досягнення: виграти 10 змагальних матчів;
   - achieved: 1 – статус досягнення: отримано (1 = отримано, 0 = не отримано);
   - unlocktime: 1625240400 – час отримання досягнення у форматі UNIX timestamp.

Отримані дані будуть інтегровані в систему для створення статистики користувача та подальшого пов’язування з відповідними посібниками. Користувачі матимуть змогу самостійно створювати керівництва, що стосуються гри або досягнень. Посібники для гри можуть містити загальну інформацію, наприклад, основні механіки гри чи способи оптимізації часу для виконання певних дій. У свою чергу, посібники для досягнень повинні

включати рекомендації та покрокові інструкції щодо їх отримання. У таких посібниках можна використовувати текст, зображення, символи та посилання на відео.

2.1.2 Рекомендації ігор та статистика популярних ігор

Принцип роботи мудуля рекомендацій наступний:

- додаток збирає дані про ігри, в які користувач грав або планує грати, а також інформацію про популярні ігри через steam арі або інші платформи;
- на основі ігрових вподобань, стилю гри, часу, проведеного в певних жанрах, та інших параметрів, додаток формує рекомендації ігор для користувача;
- статистика популярних ігор може будуватися на базі зовнішніх даних (через арі ігрових платформ), що показують тренди, рейтинг популярності та відгуки інших гравців.

Алгоритм роботи:

- відстеження ігрових жанрів та аналіз типових сесій користувача;
- використання алгоритмів рекомендацій на основі колаборативної фільтрації або простих підходів (грали користувачі з аналогічним стилем).

Рекомендації ігор у додатку будуть працювати на основі алгоритму колаборативної фільтрації. Це метод прогнозування, який ґрунтується на аналізі вподобань схожих груп користувачів. Суть методу полягає в тому, що якщо кілька користувачів мають схожі смаки, їм можуть рекомендувати контент, який сподобався іншим користувачам з тієї ж групи. Такий підхід активно використовується великими платформами, які пропонують контент, наприклад, Netflix і Spotify.

Алгоритм працює наступним чином: якщо, наприклад, три користувачі схвалили три однакові ігри, то користувач, який любить дві з цих ігор, швидше

за все зацікавиться і третьою (рис. 2.1). На основі цього припущення система пропонує йому цю гру, спираючись на дані про вподобання інших користувачів.

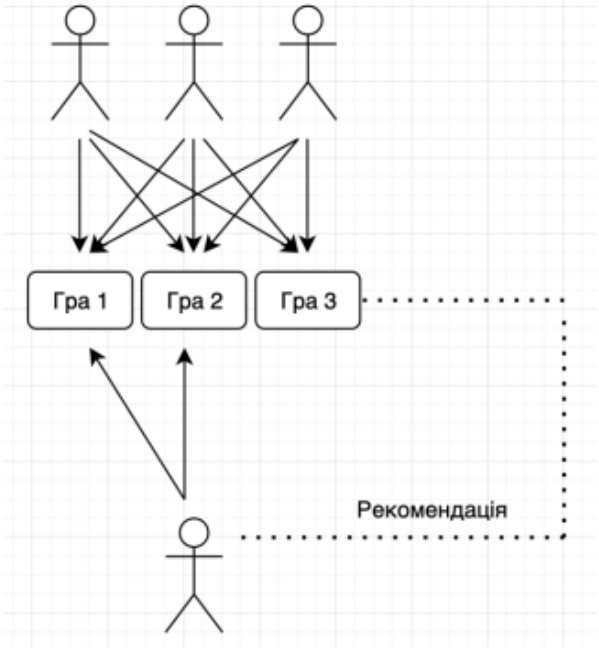

Рисунок 2.1 – Приклад роботи алгоритму коларобативної фільтрації

Троє гравців позитивно оцінили гру 1, гру 2 і гру 3. Четвертий гравець також високо оцінив перші дві гри, але ще не стикався з третьою. Алгоритм виявляє схожість у вподобаннях четвертого гравця з цими трьома користувачами. Оскільки всі троє також схвалили гру 3, система пропонує її четвертому гравцеві як таку, що може його зацікавити.Як це працює в контексті ігрового трекера:

1. Додаток аналізує ігри, в які грає користувач, і порівнює його вподобання з іншими гравцями, що мають схожі профілі.
2. Якщо система виявляє спільні ігри серед гравців, вона робить висновок, що інші ігри з цього списку також можуть бути цікавими для користувача.

3. Окрім цього, алгоритм постійно навчається на основі нових даних, удосконалюючи свої рекомендації в міру того, як користувач продовжує взаємодіяти з новими іграми.

На основі суб'єктивних суджень великої кількості людей про якість чогось формуються окремі спільноти однодумців. Усередині цих груп налагоджується постійний обмін інформацією, а довіра до нових даних базується на достовірності інформації, яка надходила раніше.

Якщо хтось із учасників спільноти першим дізнається нову інформацію, він повідомляє про це інших однодумців. У результаті, іншим учасникам вже не потрібно витрачати свої ресурси на перевірку відомостей, і вони можуть зосередитися на відкриттях для спільного блага.

Недоліком такого алгоритму є схильність до створення інформаційної бульбашки. Оскільки рекомендації та обмін інформацією відбуваються лише всередині спільноти однодумців, це може обмежувати доступ до нових або різноманітних поглядів, ідей чи продуктів. Учасники можуть отримувати лише ті дані, які вже підтверджені іншими членами групи, і це може призвести до обмеження інновацій або до того, що нова, але корисна інформація з інших джерел не буде розглянута. Також можливе накопичення помилок, якщо початкова інформація виявиться неправдивою або неточною, адже система довіряє попереднім даним.

2.1.3 Відстеження досягнень

На цьому етапі додаток інтегрується з Steam API (або аналогічними API PlayStation та Xbox) для отримання списку досягнень для кожної гри. Це дозволяє автоматично отримувати інформацію про:

- які досягнення вже були отримані;
- які досягнення ще доступні для завершення;
- окрім автоматичного відстеження, додаток може показувати деталі про досягнення (опис, умови отримання).

API:

- Steam API надає доступ до інформації про досягнення через спеціальні ендпойнти, які повертають деталі про прогрес кожного користувача;
- додаток запитує дані в реальному часі або з певною періодичністю, щоб постійно оновлювати інформацію про прогрес користувача;
- інтеграція з API дозволяє додатку також повідомляти користувача про нові досягнення, які стали доступними після оновлень гри, або про прогрес у досягненні певної мети.

Додаткові функції:

1. Сповіщення: додаток може надсилати сповіщення, коли користувач близький до отримання досягнення або коли з'являються нові досягнення після оновлення гри.
2. Прогнозування прогресу: на основі аналізу часу, витраченого на гру, додаток може прогнозувати, скільки часу залишилося для отримання певних досягнень.
3. Поради та посібники: для складних досягнень додаток може автоматично пропонувати гравцю посібники або поради, які допоможуть швидше їх отримати.

Це дозволяє користувачам не тільки зручно відстежувати свій прогрес, але й ефективно планувати свої дії для досягнення нових ігрових цілей.

2.1.4 Посібники з проходження ігор та досягнень

Принцип роботи наступний: додаток може інтегруватися з відкритими базами даних або сторонніми сервісами, які надають гайди для ігор. Це може включати:

- посібники з проходження складних рівнів;
- інструкції для отримання певних досягнень;

- автоматичне відображення гайдів може ґрунтуватися на прогресі користувача: коли користувач наближається до певного досягнення, додаток пропонує інструкції.

Джерела даних: офіційні або фанатські вебсайти (наприклад, GameFAQs, Steam Community) можуть бути інтегровані через API або парсинг контенту. Крім цього, користувач також може зберігати власні нотатки або посилання на гайди для зручного доступу.

Нижче наведено детальний опис того, як реалізуються ці функції.

1. Передача даних в систему для організації статистики.

Коли користувач проходить гру, зібрані дані про його прогрес, досягнення та проведений час передаються в систему. Ці дані використовуються для побудови персональної статистики, яка включає:

- відсоток проходження гри: оновлюється після кожної сесії;
- час, проведений у грі: накопичувальний показник загального часу, витраченого на гру;
- отримані досягнення: на основі даних від Steam API або іншої платформи;
- прогнозований час завершення гри: обчислюється на основі темпу проходження.

Мета організації статистики – надати користувачеві візуальне відображення його прогресу та на основі цього пропонувати посібники та рекомендації, які можуть допомогти скоротити час проходження або досягти конкретних цілей.

2. Прив'язка до посібників.

На основі зібраної статистики система автоматично прив'язує релевантні посібники до певних частин гри чи досягнень. Наприклад: якщо користувач має 50% прогресу в грі, система може запропонувати гайд з подальших кроків або поради, як ефективно завершити гру. Для складних досягнень система може автоматично відображати інструкції з досягнень, якщо бачить, що

користувач наближається до отримання цього досягнення (на основі даних Steam API).

3. Можливість користувачам писати керівництва.

Однією з важливих соціальних функцій додатку є можливість для користувачів створювати та публікувати власні посібники з проходження гри або отримання досягнень. Це додає інтерактивності і створює спільноту гравців, які діляться досвідом.

Створення посібників користувачами:

1. Типи посібників:
   - посібники з гри загалом: інструкції щодо основних механік гри, поради щодо ефективного проходження, як скоротити час або оптимізувати ігровий процес.
   - посібники з досягнень: поради та покрокові інструкції щодо отримання конкретних досягнень, які можуть бути складними або вимагати особливих стратегій.
2. Можливості редагування:
   - текстові інструкції: користувачі можуть створювати текстові поради та інструкції;
   - картинки та схеми: для наочності користувачі можуть додавати скріншоти з гри, схеми або інші графічні елементи, щоб допомогти іншим гравцям краще зрозуміти контекст;
   - посилання на відео: користувачі можуть вставляти посилання на відео (наприклад, з youtube), що демонструють проходження певного етапу гри або отримання досягнень;
   - символи та іконки: можна використовувати спеціальні символи, щоб виділити важливі моменти або створити структуровані поради.
3. Структура посібників:
   - заголовок: коротка назва, яка описує суть посібника (наприклад, "як отримати досягнення 'майстер стратегії'");

- опис: короткий опис того, що гравець отримає, виконуючи інструкції;
- кроки та поради: покрокові інструкції з виконання конкретних дій у грі або досягнень;
- додаткові матеріали: картинки, відео, іконки, що пояснюють конкретні етапи.

Важливо мати можливість модерувати контент та перевіряти посібники на відповідність стандартам спільноти, щоб уникати некоректної або шкідливої інформації. Можливо додати систему оцінок і коментарів для кожного посібника, щоб користувачі могли визначати, які посібники були корисні.

4. Посібники для досягнень.

Для складних і специфічних досягнень користувачі можуть створювати спеціальні посібники, які включатимуть:

- опис досягнення: що потрібно зробити, щоб його отримати;
- поради та трюки: покрокові інструкції або альтернативні шляхи для спрощення виконання завдання;
- приклади і відео: вставка відеоматеріалів або зображень для демонстрації виконання;
- інформація про час: рекомендації, скільки часу приблизно може зайняти отримання досягнення, і чи можна скоротити цей час.

5. Використання символів, картинок, посилань на відео.

Для покращення сприйняття інформації користувачі можуть використовувати мультимедійні елементи у своїх посібниках: символи та емодзі: можуть використовуватися для виділення важливих моментів, структурування кроків (наприклад, ✅ для позначення виконаних кроків).

Картинки та скріншоти: можуть використовуватися для показу локацій, ігрових елементів або тактик. Наприклад, скріншот з мапою, де відмічено місце для виконання певного завдання. Посилання на відео: користувач може

вставити посилання на відео (наприклад, YouTube), де детально показано процес виконання завдання чи проходження складного рівня.

Це додає можливість користувачам глибше зануритися в процес допомоги іншим гравцям, роблячи платформу не лише інструментом відстеження прогресу, а й джерелом навчання та обміну досвідом.

2.2 Моделювання Use-Case

Діаграма варіантів використання (Use-Case Diagram) є важливим інструментом у моделюванні вимог до системи. Вона візуалізує, як різні актори взаємодіють з системою і які функції (варіанти використання) доступні кожному з них. На рис. 2.2 представлено діаграму варіантів використання для геймтрекера, яка враховує трьох акторів: Гравець, Просунутий гравець та Модератор.

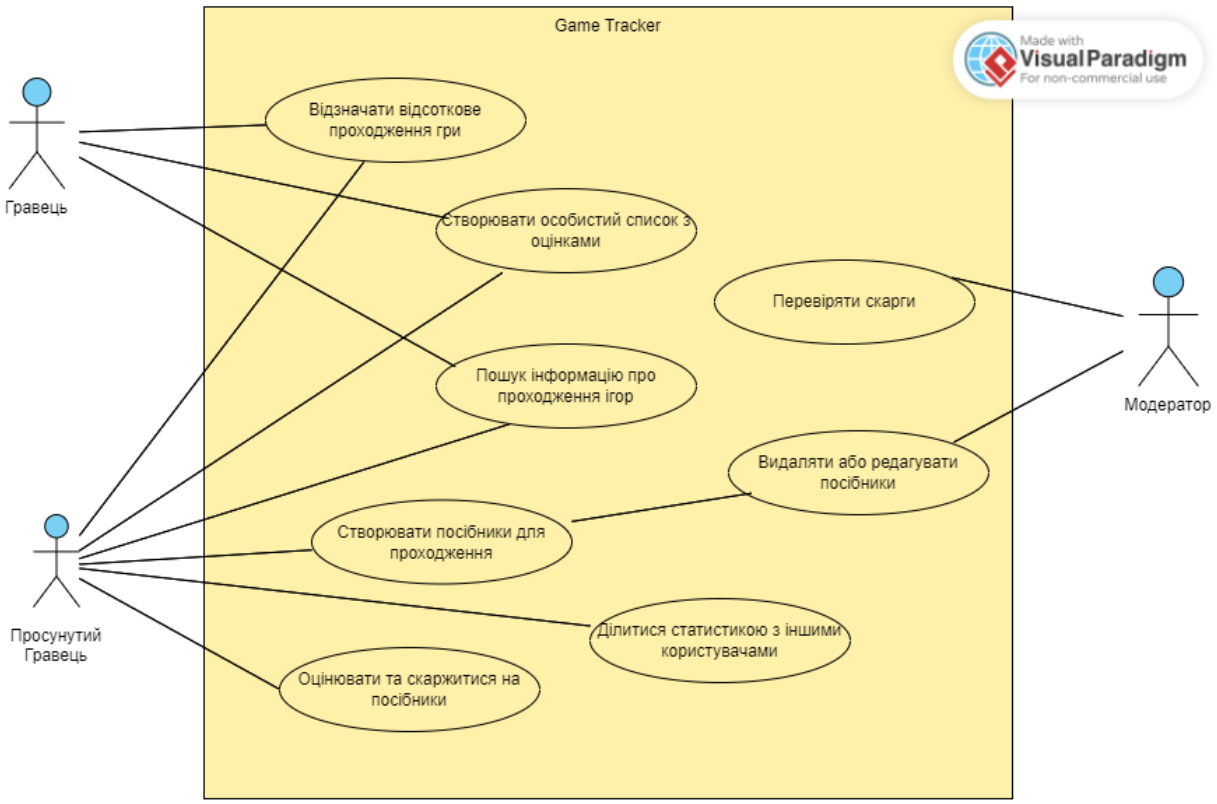

Рисунок 2.2 – Діаграма варіантів використання

Актори:

- Гравець – зареєстрований користувач з базовими функціями;
- Просунутий гравець – зареєстрований користувач з розширеними функціями;
- Модератор – користувач, відповідальний за перевірку скарг.

Кожен актор має свої варіанти використання, які ілюструють, що саме може робити кожен тип користувача в системі. Гравець має доступ до базових функцій, тоді як просунутий гравець має додаткові можливості, а модератор відповідає за підтримку якості контенту.

ВВ №1 «Відзначати відсоткове проходження гри».

Актор: Гравець.

Мета: вказати, наскільки далеко він пройшов у грі, щоб зберегти прогрес.

Основний сценарій:

1. Гравець відкриває додаток.
2. Вибирає гру зі списку.
3. Вводить відсоток проходження.
4. Зберігає зміни.
5. Додаток оновлює інформацію про прогрес.

Альтернативний сценарій: гравець помилково вводить неправильний відсоток і може редагувати його до збереження.

ВВ №2 «Створювати особистий список з оцінками».

Актор: Гравець.

Мета: зберегти улюблені ігри з оцінками для майбутнього використання.

Основний сценарій:

1. Гравець відкриває список ігор.
2. Вибирає гру, яку хоче оцінити.
3. Вводить оцінку та коментар (за бажанням).
4. Зберігає оцінку.
5. Додаток оновлює особистий список.

Альтернативний сценарій: гравець може видалити гру з особистого списку або змінити оцінку.

ВВ №3 «Читати та шукати інформацію про проходження ігор».

Актор: Гравець.

Мета: отримати доступ до корисної інформації про ігри.

Основний сценарій:

1. Гравець відкриває розділ «Посібники».
2. Використовує функцію пошуку для введення назви гри.
3. Вибирає потрібний посібник зі списку результатів.
4. Читає інформацію про проходження.

Альтернативний сценарій: гравець не знаходить жодної інформації та отримує пропозицію додати новий посібник.

ВВ №4 «Створювати посібники для проходження».

Актор: Просунутий гравець.

Мета: створити нові посібники, щоб поділитися своїм досвідом з іншими гравцями.

Основний сценарій:

1. Просунутий гравець відкриває розділ «Створити посібник».
2. Вводить назву та опис гри.
3. Додає покрокову інформацію та поради.
4. Зберігає посібник.
5. Посібник стає доступним для інших користувачів.

Альтернативний сценарій: просунутий гравець може скасувати створення посібника в будь-який момент до збереження.

ВВ №5 «Оцінювати та скаржитися на посібники».

Актор: Просунутий гравець.

Мета: забезпечити якість контенту, оцінюючи та подаючи скарги на посібники.

Основний сценарій:

1. Просунутий гравець переглядає посібники.

2. Вибирає посібник для оцінювання.
3. Вводить оцінку та залишає коментар.
4. Якщо потрібно, подає скаргу на посібник.
5. Додаток зберігає оцінку та скаргу.

Альтернативний сценарій: просунутий гравець може змінити оцінку або відкликати скаргу.

ВВ №6 «Ділитися статистикою з іншими користувачами».

Актор: Просунутий гравець.

Мета: Показати досягнення та прогрес іншим користувачам.

Основний сценарій:

1. Просунутий гравець відкриває розділ «Статистика».
2. Вибирає, що саме він хоче поділитися (досягнення, прогрес тощо).
3. Натискає кнопку «Поділитися».
4. Статистика стає видимою для інших користувачів у їхніх стрічках.

Альтернативний сценарій: просунутий гравець може скасувати публікацію статистики перед підтвердженням.

ВВ №7 «Перевіряти скарги».

Актор: Модератор.

Мета: оцінити обґрунтованість скарг, поданих просунутими гравцями.

Основний сценарій:

1. Модератор отримує повідомлення про нові скарги.
2. Переглядає деталі скарги та посібник, на який подана скарга.
3. Приймає рішення про подальші дії (додати коментар, видалити посібник тощо).

Альтернативний сценарій:

1. Отримання скарги:
   - модератор отримує повідомлення про нову скаргу на посібник, подану просунутим гравцем;

- система надає модератору деталі скарги, включаючи ідентифікаційний номер посібника, назву гри, а також текст скарги.

2. Перегляд посібника:
   - модератор відкриває посібник, на який подана скарга;
   - він вивчає зміст посібника, звертаючи увагу на якість написання (ясність, логічність), достовірність інформації (чи відповідає вона дійсності) та відповідність скарзі (наскільки обґрунтованою є скарга).
3. Пошук додаткової інформації: якщо модератор вважає, що необхідна додаткова інформація, він може:
   - переглянути інші посібники: модератор шукає подібні посібники на цю ж гру або на аналогічні теми, щоб порівняти інформацію;
   - звернутися до користувачів: модератор може зв'язатися з автором посібника або іншими користувачами, які коментували його, щоб отримати їхню думку чи пояснення;
   - консультуватися з іншими модераторами: як зазначалося в попередньому сценарії, модератор може залучити колег до обговорення, щоб отримати додаткові думки та рекомендації.
4. Оцінка інформації:
   - після збору додаткової інформації модератор повертається до первісної скарги;
   - він оцінює, чи є скарга обґрунтованою на основі зібраних даних та свого первісного аналізу.
5. Прийняття рішення:
   - якщо скарга виявилася обґрунтованою: модератор може редагувати посібник або видалити його, якщо це необхідно;
   - якщо скарга виявилася необґрунтованою: модератор може залишити посібник без змін, а також повідомити користувача, що його скарга не була підтверджена.

6. Документація дій: модератор документує результати своєї перевірки, зазначаючи:
   - причини прийнятого рішення (редагування, видалення чи залишення без змін);
   - додаткову інформацію, яка була врахована під час перевірки.
7. Повідомлення користувачів: модератор може надіслати повідомлення просунутому гравцеві, що подав скаргу, пояснюючи результати перевірки та дії, які були вжиті.

ВВ №8 «Видаляти або редагувати посібники».

Актор: Модератор.

Мета: підтримувати якість контенту, видаляючи або редагуючи ненадійні посібники.

Основний сценарій:

1. Модератор знаходить посібник, що потребує редагування чи видалення.
2. Визначає, які зміни потрібно внести.
3. Вносить редагування або видаляє посібник.
4. Додаток оновлює інформацію в системі.

Альтернативний сценарій:

1. Визначення потреби у обговоренні: модератор переглядає посібник і виявляє невизначеність щодо того, чи слід його редагувати чи видаляти. Це може бути пов’язано з неоднозначністю скарги або тим, що посібник містить частково корисну інформацію.
2. Залучення інших модераторів:
   - модератор використовує функцію внутрішнього чату або форуму, щоб зв’язатися з іншими модераторами;
   - модератор формулює запит на обговорення, детально описуючи ситуацію, включаючи назву посібника, причину скарги та власні спостереження та думки щодо посібника.

3. Обговорення:
   - інші модератори ознайомлюються з ситуацією та можуть висловлювати свої думки, давати рекомендації або ділитися подібним досвідом з іншими посібниками;
   - модератори можуть використовувати голосування або консенсус, щоб визначити, чи слід редагувати чи видаляти посібник.
4. Прийняття рішення: після обговорення модератор (або група модераторів) приймає рішення про подальші дії:
   - якщо вирішено редагувати посібник, модератор повертається до етапу редагування, враховуючи рекомендації;
   - якщо вирішено видалити посібник, модератор переходить до процедури видалення.
5. Документація дій: модератор документує результати обговорення та прийняті рішення для подальшої звітності. Це може включати:
   - запис причин редагування чи видалення;
   - залишення коментаря під посібником (якщо редагування відбулося), щоб пояснити користувачам, чому внесено зміни.
6. Повідомлення користувачів: якщо посібник було видалено, модератор може надіслати повідомлення, щоб проінформувати користувачів про видалення та причини, щоб підтримати прозорість.

2.3 Проектування архітектури системи

Розробка архітектури для геймтрекеру передбачає врахування різних компонентів системи, які забезпечать її ефективність, масштабованість і надійність. Архітектура Model-View-Controller (MVC) є популярним патерном проектування, який розділяє додаток на три основні компоненти: модель, представлення та контролер. Відповідно до геймтрекеру архітектура буде мати наступну структуру (рис. 2.3).

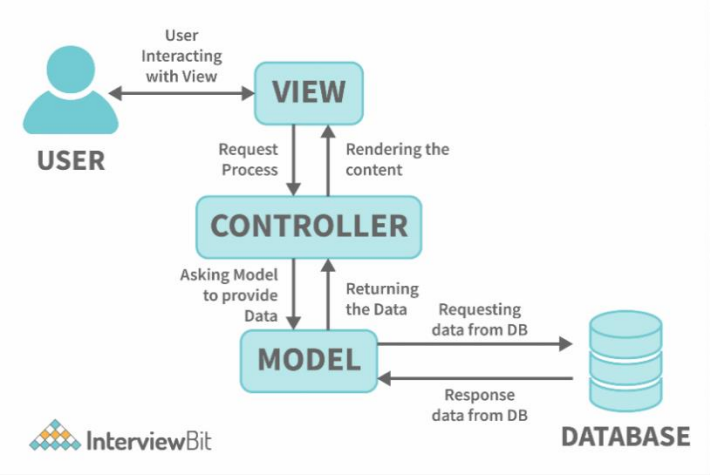

Рисунок 2.2 – Архітектура MVC

Нижче наведено опис до кожного компонента моделі.

1. Model (Модель) відповідає за управління даними, логікою бізнесу та правилами програми. У контексті геймтрекера модель може включати:

- клас користувача: зберігає інформацію про зареєстрованих користувачів (гравців, просунутих гравців, модераторів), їхні профілі та налаштування;
- клас гри: зберігає дані про ігри (назва, жанр, опис, доступні досягнення);
- клас досягнень: включає інформацію про досягнення для кожної гри (назва, опис, статус);
- клас статистики: зберігає дані про час, проведений у грі, відсоток проходження, рейтинг ігор;
- клас посібників: включає дані про створені посібники, їхній зміст, авторів та оцінки.

2. View (Представлення) відповідає за відображення даних, що надходять від моделі, і взаємодію з користувачем. У геймтрекері представлення включає:

- головний інтерфейс користувача: відображає меню, кнопки, панелі навігації та інші елементи;

- сторінки профілю: відображають інформацію про користувача, його досягнення та прогрес у іграх;
- сторінка статистики: відображає графіки та таблиці з даними про час гри, досягнення, рейтинг тощо;
- сторінка посібників: відображає список доступних посібників, їхні деталі, а також форму для створення нового посібника;
- сторінка скарг: відображає інформацію про скарги на посібники, а також форму для подання нової скарги.

3. Controller (Контролер) працює посередником між моделлю і представленням. Він обробляє введення користувача, виконує відповідні дії з моделлю і оновлює представлення. У геймтрекері контролер включає:

- UserController: обробляє аутентифікацію користувачів, реєстрацію, оновлення профілю, а також запити на отримання даних про користувача;
- GameController: управляє логікою взаємодії з іграми, включаючи отримання списку ігор, додавання нових ігор, а також оновлення інформації про досягнення;
- AchievementController: обробляє запити на отримання досягнень, оновлення статусу досягнень, а також взаємодію з зовнішніми API для отримання інформації про досягнення;
- GuideController: відповідає за обробку запитів на створення, редагування та видалення посібників, а також управління оцінками і коментарями до посібників;
- ComplaintController: обробляє скарги на посібники, проводить їх перевірку, а також управляє комунікацією з користувачами, що подали скарги.

Взаємодія між компонентами MVC:

1. Введення користувача: користувач взаємодіє з представленням (наприклад, заповнює форму для створення посібника).

2. Обробка контролером: представлення надсилає запит до контролера (наприклад, GuideController), який обробляє введені дані.
3. Зміни в моделі: контролер взаємодіє з моделлю для виконання дій (наприклад, створення нового посібника).
4. Оновлення представлення: модель оновлює дані в базі даних, а контролер повідомляє представлення про зміни, щоб оновити інтерфейс користувача.

Для демонстрації, як користувачі будуть взаємодіяти з геймтрекером на рис. 2.3 наведено діаграму станів.

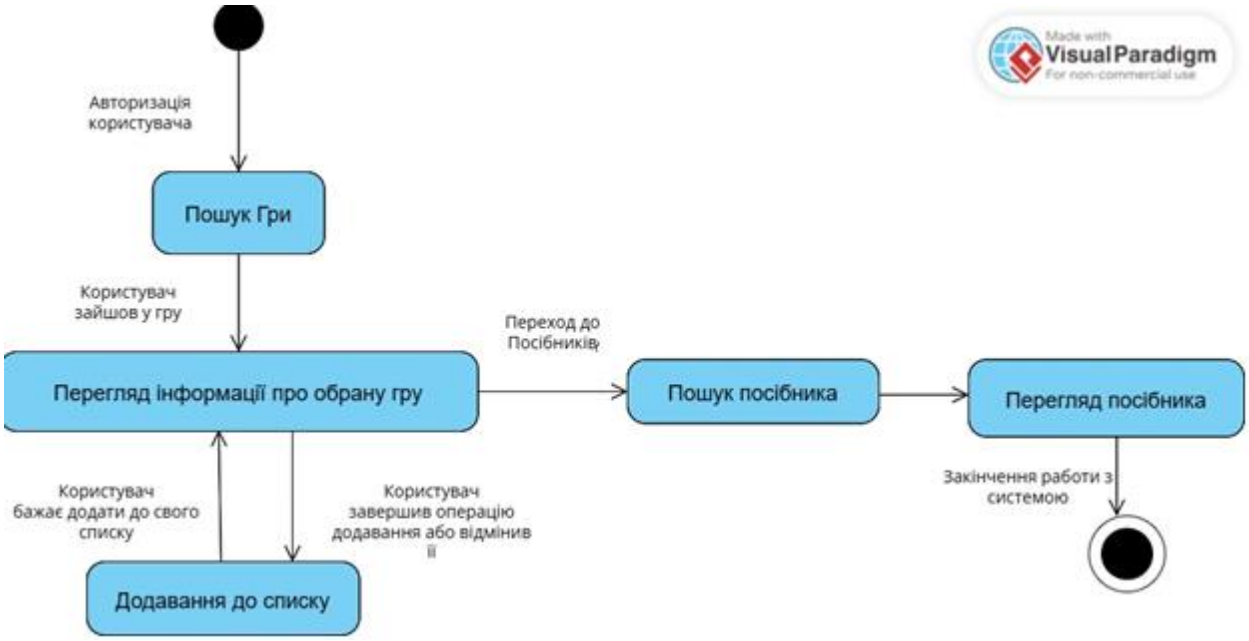

Рисунок 2.3 – Діаграма станів роботи з веб-додатком

Основни елементами тут є взаємодії, рішення та ситуації.

Взаємодії: діаграма ілюструє, як користувач взаємодіє з додатком, проходячи через різні етапи пошуку та перегляду інформації.

Рішення: діаграма показує, які дії користувач може виконати в різних ситуаціях (додавання до списку, перегляд посібників).

Процес: описує послідовність дій, які ведуть до досягнення мети користувача.

Опис діаграми:

1. Вхід до додатку: користувач заходить у додаток, що є початковою точкою взаємодії.
2. Пошук гри: після входу користувач має можливість виконати пошук гри. Це може бути реалізовано через пошуковий інтерфейс, де користувач вводить назву гри.
3. Перегляд інформації про гру: якщо користувач знайшов гру, він може переглянути детальну інформацію про неї, включаючи опис, досягнення, рейтинги тощо.
4. Додавання до списку:
   - якщо користувач вирішує додати гру до свого особистого списку, він переходить до відповідної дії;
   - відбувається процес додавання гри до списку, і в результаті гра зберігається в обліковому записі користувача.
5. Пошук посібника: після перегляду інформації про гру користувач може також перейти до пошуку посібника, щоб знайти допомогу або поради щодо проходження гри.
6. Перегляд посібника: користувач має можливість переглянути знайдений посібник, щоб ознайомитися з корисною інформацією та порадами.
7. Завершення роботи з системою: Користувач може завершити свою сесію в системі, що є фінальним етапом у цій діаграмі.

2.4 Моделювання режимів роботи у веб-додатку

Діаграми діяльності використовуються для моделювання динаміки системи, зображаючи потоки контролю або даних в процесі виконання певних дій або сценаріїв. Розглянемо роботу користувача з пошуком ігор на рис. 2.4.

Ця діаграма допомагає зрозуміти, як користувач може здійснити пошук у системі, а також які етапи проходить його запит.

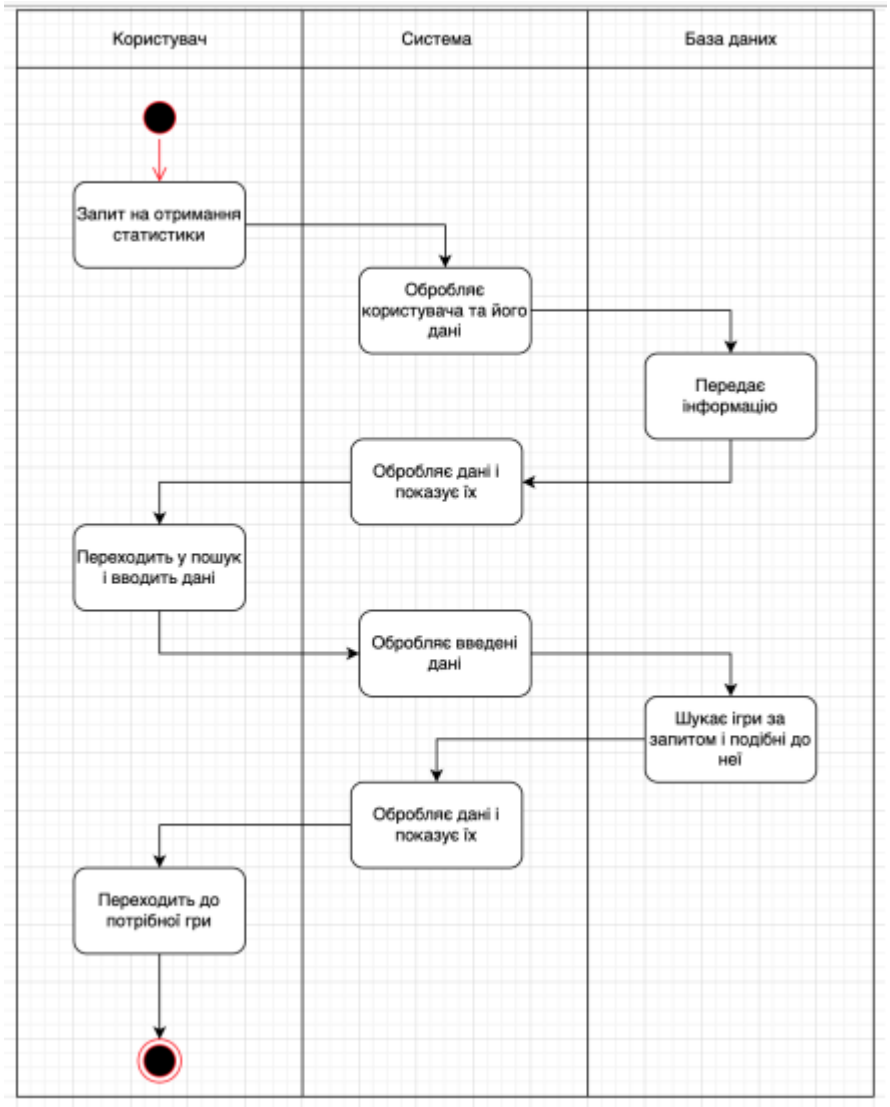

Рисунок 2.3 – Діаграма діяльності для пошуку ігор

Діаграма починається з того, що користувач ініціює запит на отримання статистики. Це може бути зроблено через інтерфейс користувача, де він вказує, які саме статистичні дані його цікавлять. Система обробляє запит, отримуючи дані користувача. Це може включати інформацію про його обліковий запис, збережені ігри та інші налаштування.

Після обробки даних система передає необхідну інформацію, щоб підтвердити запит користувача. Користувач переходить у пошук і вводить дані, такі як назва гри або критерії для пошуку (жанр, рейтинг тощо). Система обробляє введені дані і показує список ігор, що відповідають запиту користувача. Далі система виконує пошук гри за введеними критеріями і надає результати, дозволяючи користувачеві переглянути і вибрати потрібну гру.

На рис. 2.4 представлено роботу користувача з посібниками. На першому кроці користувач заходить у веб-додаток, використовуючи свої облікові дані (логін та пароль). Після успішної аутентифікації система автоматично відображає інформацію про останні сесії користувача, включаючи статистику прогресу в іграх, досягнення, які були отримані, та посібники, які він переглядав.

Далі користувач переглядає свій список ігор і обирає ту, по якій хоче отримати більше інформації. Система обробляє запит та передає користувачеві всі дані, пов’язані з обраною грою, включаючи:

- назву гри;
- загальний прогрес користувача у цій грі;
- список досягнень, які ще не отримані;
- доступні посібники для проходження гри.

Користувач переходить до вкладки, де представлені всі посібники для обраної гри. Користувач може скористатися функцією фільтрації або пошуку, щоб знайти конкретний посібник, який йому потрібен. Після перегляду списку посібників користувач обирає той, який його цікавить. Система обробляє запит на відкриття вибраного посібника.

Система передає користувачеві всю інформацію, пов’язану з вибраним посібником(текстовий опис з покроковими інструкціями, знімок екрана, який демонструє важливі елементи або моменти гри, відео, що ілюструє процес проходження або показує складні частини гри). Користувач може переглядати текст, знімки та відео в зручному для себе форматі, користуючись вбудованими елементами управління (прокрутка, відтворення відео тощо).

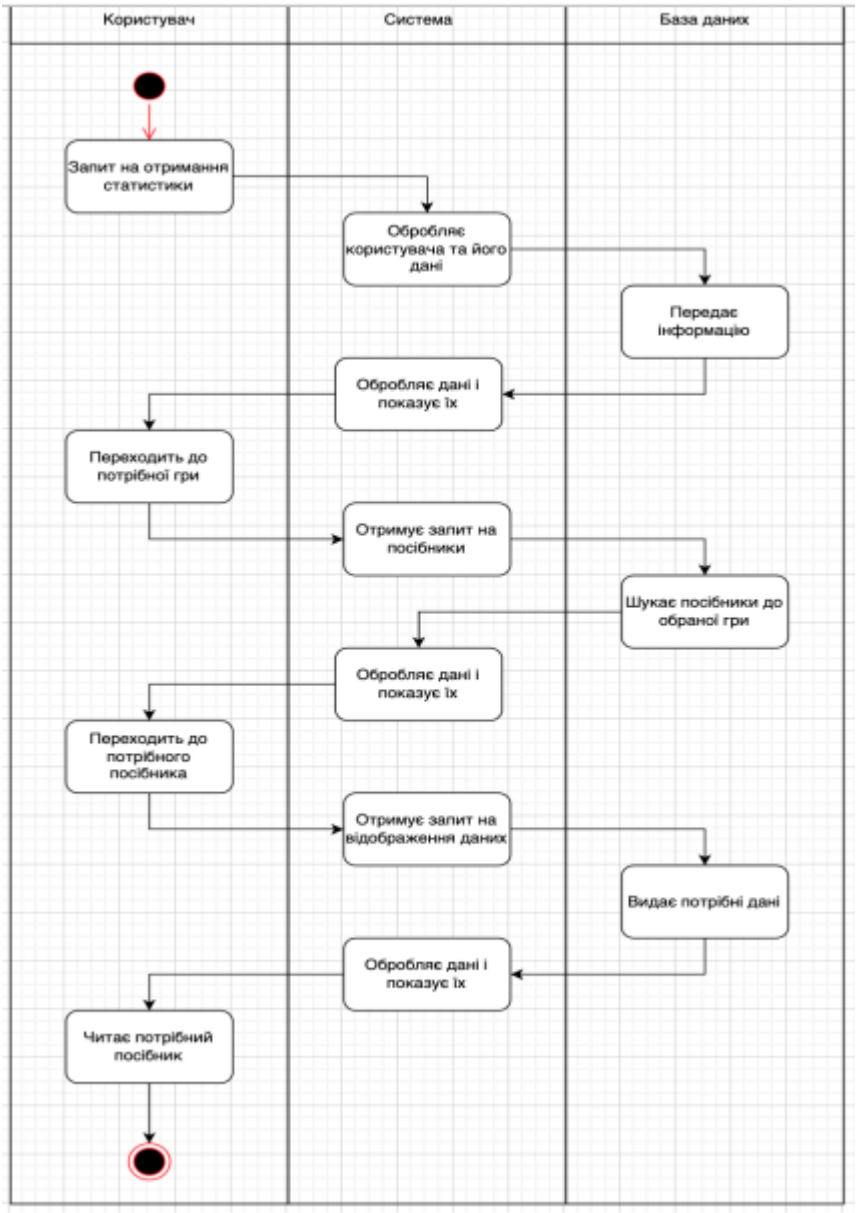

Рисунок 2.4 – Діаграма діяльності по роботі з посібниками

Після того, як користувач ознайомиться з інформацією, він може вирішити повернутися до списку посібників, перейти до інших вкладок, таких як статистика або загальна інформація про гру, або вийти з додатку. Якщо користувач вирішить, що посібник був корисним, він може залишити відгук або оцінку, що сприятиме поліпшенню контенту у майбутньому.

3 РОЗРОБКА ВЕБ-ЗАСТОСУНКУ

3.1 Обгрунтування вибору засобів розробки

Для реалізації проекту геймтрекера можна обрати різні технології та інструменти в залежності від архітектури, функціональних вимог та цілей. Вибір пав на наступні:

1. Для Frontend (Клієнтської частини): React, CSS-фреймворк Bootstrap.
2. Для Backend (Серверної частини): Node.js, Express.js, GraphQL.
3. База даних: PostgreSQL.
4. Засоб для інтеграції з API (Steam, PlayStation, Xbox): Axios.
5. Система автентифікації та авторизації: JWT (JSON Web Tokens).

React – це бібліотека JavaScript, розроблена Facebook, яка використовується для створення інтерфейсів користувача (UI) для веб-додатків. Основна мета React – спростити процес розробки інтерфейсів, роблячи їх більш продуктивними та легкими у підтримці.

Основні особливості:

1. Компонентний підхід: React дозволяє розбивати UI на окремі, повторно використовувані компоненти. Кожен компонент може мати власний стан та поведінку, що спрощує управління складними інтерфейсами.
2. Virtual DOM: React використовує концепцію Virtual DOM, що дозволяє оптимізувати оновлення UI. Замість прямого маніпулювання DOM, React спочатку оновлює Virtual DOM, а потім порівнює його з реальним DOM, застосовуючи тільки необхідні зміни.
3. Однонаправлений потік даних: у React дані передаються лише в одному напрямку, з батьківських компонентів до дочірніх. Це спрощує відстеження змін і підвищує контроль над станом додатка.

4. JSX: React використовує JSX – синтаксис, який дозволяє писати HTML-подібний код в JavaScript. Це робить код більш зрозумілим та читабельним [7].

Bootstrap є популярним CSS-фреймвором, який спрощує розробку адаптивних веб-додатків. Розроблений Twitter, Bootstrap пропонує набір готових компонентів та стилів, що дозволяє швидко створювати привабливі та функціональні інтерфейси.

Основні особливості:

1. Адаптивний дизайн: Bootstrap підтримує адаптивний (responsive) дизайн, що дозволяє веб-додаткам автоматично підлаштовуватися під різні розміри екранів і пристроїв.
2. Готові компоненти: Bootstrap надає безліч готових UI-компонентів, таких як кнопки, форми, навігаційні панелі, модальні вікна та багато інших, які можна легко налаштовувати.
3. Система сіток: Bootstrap використовує систему сіток (grid system), яка дозволяє легко організовувати контент на сторінці. Це забезпечує гнучке розміщення елементів і оптимізує простір на екрані.
4. Кастомізація: Bootstrap дозволяє користувачам налаштовувати стилі за допомогою Sass або CSS, щоб відповідати вимогам конкретного проекту. Користувачі можуть змінювати кольори, шрифти, розміри елементів та інші стилі.
5. Документація: Bootstrap має детальну документацію, що полегшує навчання та використання фреймворку. Вона містить приклади коду, пояснення компонентів і налаштувань [8].

React буде використано для створення динамічного та інтерактивного інтерфейсу користувача геймтрекера. Завдяки компонентному підходу, можна легко управляти станом програми та реалізовувати складні функціональні можливості. Bootstrap допоможе швидко створити привабливий і адаптивний дизайн. Використання готових компонентів з Bootstrap зменшить час розробки та дозволить зосередитися на функціональності додатку.

Node.js – це серверна платформа, побудована на основі JavaScript, яка дозволяє розробникам створювати швидкі та масштабовані веб-додатки. Вона використовує подієву, асинхронну модель вводу/виводу, що робить її особливо підходящою для створення високонавантажених додатків, які обробляють багато одночасних запитів.

Node.js використовує однопотокову модель з подієвим циклом, що дозволяє обробляти багато запитів без блокування, забезпечуючи високу продуктивність. Вона має потужний менеджер пакетів (NPM), який надає доступ до тисяч бібліотек і модулів, що можна використовувати для розширення функціональності додатка. Крім цього, Node.js дозволяє використовувати JavaScript як на клієнтській, так і на серверній стороні, що спрощує розробку та зменшує потребу в різних мовах програмування. Node.js ідеально підходить для створення RESTful API, веб-серверів, чат-додатків, реальних часів (real-time) додатків та багатьох інших [9].

Express.js – це легкий та гнучкий веб-фреймворк для Node.js, який спрощує створення веб-додатків і API. Він надає простий і зрозумілий інтерфейс для роботи з HTTP-запитами і маршрутизацією. Express.js надає мінімалістичний і простий у використанні інтерфейс для створення серверних додатків. Розробники можуть швидко налаштувати сервер і визначити маршрути.

Завдяки своїй простоті та модульній архітектурі Express.js легко розширюється, що дозволяє додавати нові функції за допомогою middleware. Він підтримує концепцію middleware, що дозволяє вставляти функції, які можуть обробляти запити перед тим, як вони досягнуть кінцевих обробників. Це дозволяє реалізувати функціонал, такий як аутентифікація, логування та обробка помилок. Express.js підтримує різні шаблонізатори (таких як EJS, Pug), що дозволяє генерувати HTML-сторінки на сервері.

GraphQL – це мова запитів для API, розроблена Facebook, яка дозволяє клієнтам запитувати лише ті дані, які їм потрібні. Це забезпечує більшу гнучкість і ефективність порівняно з традиційними REST API.

Клієнти можуть формулювати свої запити так, щоб отримувати лише необхідні дані, що зменшує обсяг переданих даних і підвищує продуктивність. GraphQL дозволяє об'єднувати кілька запитів в один, що спрощує отримання даних з різних ресурсів. Він використовує систему типів, що дозволяє точно визначати структуру даних і забезпечує валідацію запитів перед їх виконанням. Крім цього, GraphQL дозволяє додавати нові поля до API без ризику порушити існуючі запити, що спрощує розвиток API.

Axios – це популярна бібліотека для виконання HTTP-запитів у JavaScript. Вона працює як у браузері, так і в середовищі Node.js. Axios є обгорткою для стандартного методу XMLHttpRequest і надає зручний інтерфейс для роботи з HTTP-запитами, що спрощує обробку асинхронних запитів до серверів.

Axios буде використовуватися для виконання запитів до вашого серверного API, створеного за допомогою Node.js та Express.js. Це дозволить клієнтській частині (React) отримувати і відправляти дані на сервер. Axios спростить обробку відповідей сервера, автоматично перетворюючи їх у формат JSON і дозволяючи легко обробляти отримані дані. Завдяки перехоплювачам, ви зможете реалізувати логіку для обробки запитів та відповідей, що покращить ваш досвід роботи з API та допоможе в дебагу [10].

3.2 Розробка БД

За результатами аналізу предметної області можна виділити наступні сутності для бази даних:

1. Таблиця «Користувачі» (Users):
   - user_id (PK) – унікальний ідентифікатор користувача;
   - username – ім'я користувача;
   - email – електронна адреса;
   - password_hash – хеш пароля;

- user_role – роль користувача (гравець, просунутий гравець, модератор);
- created_at – дата створення облікового запису.

2. Таблиця «Ігри» (Games):
   - game_id (PK) – унікальний ідентифікатор гри;
   - title – назва гри;
   - description – опис гри;
   - release_date – дата виходу;
   - platform – платформа (Steam, PlayStation, Xbox);
   - developer – розробник гри.
3. Таблиця «Досягнення» (Achievements):
   - achievement_id (PK) – унікальний ідентифікатор досягнення;
   - game_id (FK) – зв'язок з таблицею Games;
   - title – назва досягнення;
   - description – опис досягнення;
   - points – кількість очок за досягнення.
4. Таблиця «Статистика» (Statistics):
   - statistic_id (PK) – унікальний ідентифікатор статистики;
   - user_id (FK) – зв'язок з таблицею Users;
   - game_id (FK) – зв'язок з таблицею Games;
   - completion_percentage – відсоток проходження;
   - hours_played – години, проведені в грі;
   - achievements_unlocked – кількість досягнень, що були отримані.
5. Таблиця «Посібники» (Guides):
   - guide_id (PK) – унікальний ідентифікатор посібника;
   - game_id (FK) – зв'язок з таблицею Games;
   - user_id (FK) – зв'язок з таблицею Users (автор посібника);
   - title – назва посібника;
   - content – текст посібника;
   - created_at – дата створення посібника;

- updated_at – дата останнього редагування.

6. Таблиця «Скарги» (Reports):
   - report_id (PK) – унікальний ідентифікатор скарги;
   - guide_id (FK) – зв'язок з таблицею Guides;
   - user_id (FK) – зв'язок з таблицею Users (той, хто подав скаргу);
   - reason – причина скарги;
   - status – статус скарги (очікує, розглянута, відхилена);
   - created_at – дата подання скарги.

На рис. 3.1 представлено діаграму ER.

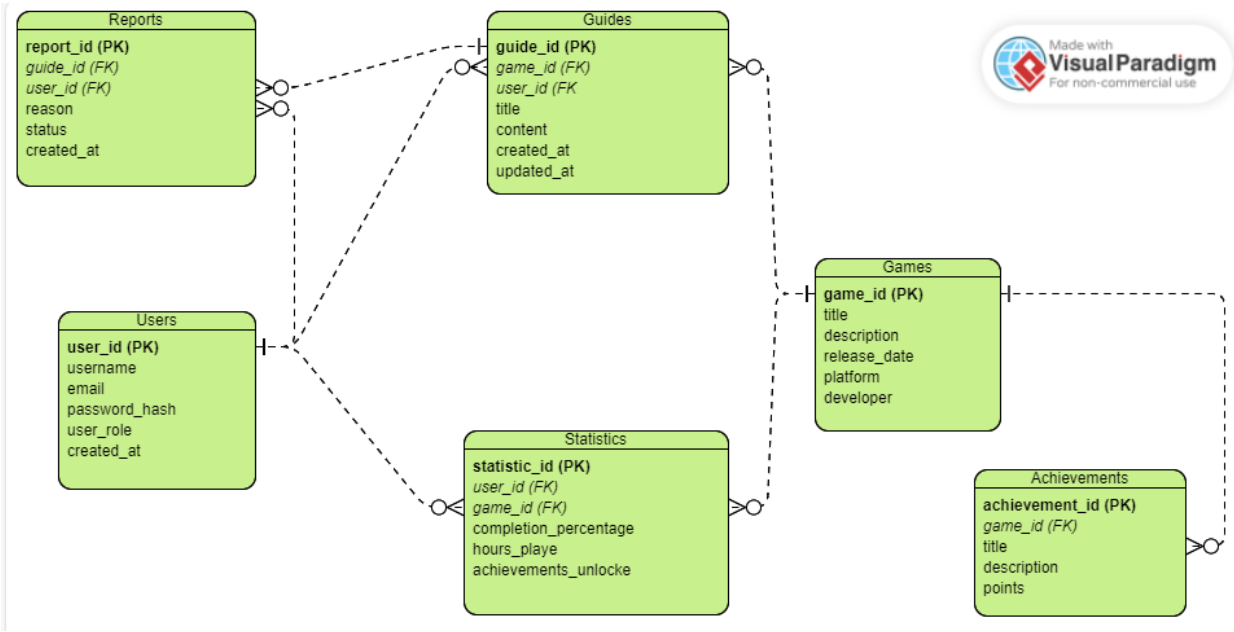

Рисунок 3.1 – Діаграма «Сутність-зв’язок»

SQL-скрипт для створення таблиць:

```
CREATE TABLE Users (
    user_id SERIAL PRIMARY KEY,
    username VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    user_role ENUM('player', 'advanced_player', 'moderator')
NOT NULL,
```

```
        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    );

    CREATE TABLE Games (
        game_id SERIAL PRIMARY KEY,
        title VARCHAR(100) NOT NULL,
        description TEXT,
        release_date DATE,
        platform VARCHAR(50),
        developer VARCHAR(100)
    );

    CREATE TABLE Achievements (
        achievement_id SERIAL PRIMARY KEY,
        game_id INT REFERENCES Games(game_id),
        title VARCHAR(100) NOT NULL,
        description TEXT,
        points INT DEFAULT 0
    );

    CREATE TABLE Statistics (
        statistic_id SERIAL PRIMARY KEY,
        user_id INT REFERENCES Users(user_id),
        game_id INT REFERENCES Games(game_id),
        completion_percentage  INT  CHECK  (completion_percentage
BETWEEN 0 AND 100),
        hours_played INT DEFAULT 0,
        achievements_unlocked INT DEFAULT 0
    );

    CREATE TABLE Guides (
        guide_id SERIAL PRIMARY KEY,
        game_id INT REFERENCES Games(game_id),
        user_id INT REFERENCES Users(user_id),
        title VARCHAR(100) NOT NULL,
        content TEXT,
        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
    );

    CREATE TABLE Reports (
        report_id SERIAL PRIMARY KEY,
        guide_id INT REFERENCES Guides(guide_id),
        user_id INT REFERENCES Users(user_id),
        reason TEXT,
```

```
        status ENUM('pending', 'resolved', 'rejected') DEFAULT
'pending',
        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    );
```

3.3 Концептуальні класи та основні функції

Реалізація концептуальних класів на JavaScript для обробки даних на сервері (Node.js) за допомогою Express.js та використання взаємодії з базою даних PostgreSQL через ORM.

Нижче наведено приклади для основних класів: «User», «Game», «Guide», «Achievement», «Recommendation».

1. User Class (Клас користувача):

```
const { Model, DataTypes } = require('sequelize');
const sequelize = require('../config/database');
class User extends Model {}
User.init({
  id: {
    type: DataTypes.UUID,
    defaultValue: DataTypes.UUIDV4,
    primaryKey: true
  },
  name: {
    type: DataTypes.STRING,
    allowNull: false
  },
  email: {
    type: DataTypes.STRING,
    allowNull: false,
    unique: true,
    validate: {
      isEmail: true
    }
  },
  password: {
    type: DataTypes.STRING,
    allowNull: false
  },
  role: {
```

```
        type: DataTypes.ENUM('player', 'advanced', 'moderator'),
        defaultValue: 'player'}},
    {
      sequelize,
      modelName: 'User',
      tableName: 'users',
      timestamps: true
    });
    module.exports = User;
```

2. Game Class (Клас гри):

```
const { Model, DataTypes } = require('sequelize');
const sequelize = require('../config/database');
class Game extends Model {}
Game.init({
  id: {
    type: DataTypes.UUID,
    defaultValue: DataTypes.UUIDV4,
    primaryKey: true
  },
  title: {
    type: DataTypes.STRING,
    allowNull: false
  },
  genre: {
    type: DataTypes.STRING,
    allowNull: true
  },
  platform: {
    type: DataTypes.STRING,
    allowNull: false
  },
  description: {
    type: DataTypes.TEXT,
    allowNull: true
  }
}, {
  sequelize,
  modelName: 'Game',
  tableName: 'games',
  timestamps: true
});
module.exports = Game;
```

3. Guide Class (Клас посібника):

```
const { Model, DataTypes } = require('sequelize');
const sequelize = require('../config/database');
const User = require('./User');
const Game = require('./Game');
class Guide extends Model {}
Guide.init({
  id: {
    type: DataTypes.UUID,
    defaultValue: DataTypes.UUIDV4,
    primaryKey: true
  },
  title: {
    type: DataTypes.STRING,
    allowNull: false
  },
  content: {
    type: DataTypes.TEXT,
    allowNull: false
  },
  media: {
    type: DataTypes.JSON,
    allowNull: true
  },
  gameId: {
    type: DataTypes.UUID,
    allowNull: false,
    references: {
      model: Game,
      key: 'id'
    }
  },
  userId: {
    type: DataTypes.UUID,
    allowNull: false,
    references: {
      model: User,
      key: 'id'
  }
}, {
  sequelize,delName: 'Guide',
  tableName: 'guides',
  timestamps: true
```

```
});
module.exports = Guide;
```

4. Achievement Class (Клас досягнень):

```
const { Model, DataTypes } = require('sequelize');
const sequelize = require('../config/database');
const User = require('./User');
const Game = require('./Game');
class Achievement extends Model {}
Achievement.init({
  id: {
    type: DataTypes.UUID,
    defaultValue: DataTypes.UUIDV4,
    primaryKey: true
  },
  name: {
    type: DataTypes.STRING,
    allowNull: false
  },
  status: {
    type: DataTypes.ENUM('achieved', 'in-progress', 'not-
started'),
    defaultValue: 'not-started',
  conditions: {
    type: DataTypes.TEXT,
    allowNull: false},
  gameId: {
    type: DataTypes.UUID,
    allowNull: false,
    references: {
      model: Game,
      key: 'id'}
  },
  userId: {
    type: DataTypes.UUID,
    allowNull: false,
    references: {
      model: User,
      key: 'id'
    }}},
{
  sequelize,
  modelName: 'Achievement',
```

```
      tableName: 'achievements',
      timestamps: true
    });
    module.exports = Achievement;
```

5. Recommendation Class (Клас рекомендацій):

```
const { Model, DataTypes } = require('sequelize');
const sequelize = require('../config/database');
const User = require('./User');
const Game = require('./Game');
class Recommendation extends Model {}
Recommendation.init({
  id: {
    type: DataTypes.UUID,
    defaultValue: DataTypes.UUIDV4,
    primaryKey: true
  },
  rating: {
    type: DataTypes.INTEGER,
    allowNull: false
  },
  gameId: {
    type: DataTypes.UUID,
    allowNull: false,
    references: {
      model: Game,
      key: 'id'}
  },
  userId: {
    type: DataTypes.UUID,
    allowNull: false,
    references: {
      model: User,
      key: 'id'}}
}, {
  sequelize,
  modelName: 'Recommendation',
  tableName: 'recommendations',
  timestamps: true
});
module.exports = Recommendation;
```

Використання цих класів із GraphQL для отримування або зміни даних (код описує, як отримувати список ігор та створювати нового користувача в системі). Спочатку слід імпортувати типи для GraphQL та моделі User, Game, Guide:

```
const { GraphQLObjectType, GraphQLString, GraphQLList,
GraphQLNonNull } = require('graphql');
const { UserType, GameType, GuideType } = require('./types');
const User = require('../models/User');
const Game = require('../models/Game');
const Guide = require('../models/Guide');
```

Далі потрібен запит для отримання всіх ігор (назва запиту «games», який повертає список всіх ігор. Визначаємо тип відповіді як список об'єктів типу GameType (він описує схему для гри)

```
const gameQueries = {
type: new GraphQLList(GameType),
```

Функція resolve виконує реальну логіку запиту. Повертаємо всі ігри з бази даних за допомогою Sequelize:

```
 resole: async () => {
 return await Game.findAll();
    }
```

Mutation для створення нового користувача. Тут визначаємо тип відповіді як UserType (схема користувача) та вказуємо аргументи, які потрібно передати для створення нового користувача:

```
const createUser = {
 type: UserType,
 args: {
    name: { type: new GraphQLNonNull(GraphQLString) },
    email: { type: new GraphQLNonNull(GraphQLString) },
```

```
    password: { type: new GraphQLNonNull(GraphQLString) }
  },
```

Функція resolve виконує логіку мутації. Використовуємо модель User для створення нового запису в базі даних. Ім'я користувача, Email та пароль передаються як аргумент у запиті:

```
 resolve: async (parent, args) => {
    return await User.create({
      name: args.name,
      email: args.email,
      password: args.password   });
  }
};
```

GraphQL-схеми для типів користувачів, ігор і посібників: тут UserType, GameType, GuideType описують GraphQL-схеми для об'єктів "користувач", "гра" та "посібник". Ці типи використовуються для визначення структури відповіді GraphQL-запитів та мутацій.

```
const { GraphQLObjectType, GraphQLString, GraphQLID, GraphQLList
} = require('graphql');
```

Тип користувача (UserType)

```
const UserType = new GraphQLObjectType({
  name: 'User',
  fields: () => ({
    id: { type: GraphQLID },
    name: { type: GraphQLString },
    email: { type: GraphQLString }
  })
});
```

Тип гри (GameType):

```
const GameType = new GraphQLObjectType({
  name: 'Game',
```

```
  fields: () => ({
    id: { type: GraphQLID },
    title: { type: GraphQLString },
    genre: { type: GraphQLString },
    platform: { type: GraphQLString }
  })
});
```

Тип посібника (GuideType):

```
const GuideType = new GraphQLObjectType({
  name: 'Guide',
  fields: () => ({
    id: { type: GraphQLID },
    title: { type: GraphQLString },
    content: { type: GraphQLString },
    media: { type: GraphQLString }
  })
});
module.exports = { UserType, GameType, GuideType };
```

3.4 Реалізація модуля статистики

Формування статистики гравця з урахуванням його досягнень на різних платформах потребує інтеграції з API таких платформ, як Steam, PlayStation Network та Xbox Live. Кожне з цих API надає дані про досягнення, які можна використовувати для збирання та обробки статистики.

Неохідно виконати три кроки:

1. Інтеграція з різними API платформ (Steam, PlayStation, Xbox).
2. Зберігання даних у базі.
3. Агрегація та обробка даних для створення статистики.

Нижче наведено концептуальний приклад коду, який реалізує цей процес:

1. Модуль для роботи з платформами.

Потрібно створити клас для роботи з Steam API (SteamService):

```
const axios = require('axios');
class SteamService {
  constructor(apiKey) {
    this.apiKey = apiKey;
    this.baseUrl = 'https://api.steampowered.com';
  }
```

Наступний крок – це отримати досягнення користувача:

```
 async getAchievements(steamId) {
    const url =
`${this.baseUrl}/ISteamUserStats/GetPlayerAchievements/v1/?key=$
{this.apiKey}&steamid=${steamId}`;
    const response = await axios.get(url);
    return response.data.playerstats.achievements;
  }
```

Аналогічно необхідно описати класи для роботи з Xbox API та з PlayStation API.

2. Сервіс для агрегації даних.

```
const { SteamService, PlayStationService, XboxService } =
require('./platformServices');
```

Основний сервіс для формування статистики

```
class PlayerStatsService {
  constructor(steamService, psService, xboxService) {
    this.steamService = steamService;
    this.psService = psService;
    this.xboxService = xboxService;
  }
```

Отримуємо загальну статистику гравця і його досягнення з усіх платформ:

```
 async getPlayerStatistics(player) {
    const { steamId, psnId, xboxId } = player;
    const steamAchievements = steamId ? await
this.steamService.getAchievements(steamId) : [];
```

```
        const psAchievements = psnId ? await
this.psService.getAchievements(psnId) : [];
        const xboxAchievements = xboxId ? await
this.xboxService.getAchievements(xboxId) : [];
```

Об'єднуємо всі досягнення в одну структуру та обчислюємо статистику:

```
    const allAchievements = [
          ...steamAchievements,
          ...psAchievements,
          ...xboxAchievements
        ];
        const totalAchievements = allAchievements.length;
        const completedAchievements = allAchievements.filter(ach
=> ach.completed).length;
        const completionRate = (completedAchievements /
totalAchievements) * 100;
        return {
          totalAchievements,
          completedAchievements,
          completionRate
        };
      }
    }
    module.exports = PlayerStatsService;
```

4. Запит даних та формування відповіді:

```
    const { SteamService, PlayStationService, XboxService } =
require('./platformServices');
    const PlayerStatsService = require('./playerStatsService');
```

Ініціалізуємо сервіси платформ з відповідними ключами доступу та створюємо основний сервіс для збору статистики:

```
    const steamService = new SteamService('STEAM_API_KEY');
    const psService = new
PlayStationService('PLAYSTATION_ACCESS_TOKEN');
    const xboxService = new XboxService('XBOX_ACCESS_TOKEN');
```

```
    const playerStatsService = new
PlayerStatsService(steamService, psService, xboxService);
```

Приклад виклику для користувача:

```
    const player = {
      steamId: '123456789',
      psnId: 'player_psn',
      xboxId: 'player_xbox'
    };
```

Формування статистики для конкретного користувача:

```
    playerStatsService.getPlayerStatistics(player)
      .then(stats => {
        console.log('Статистика гравця:', stats);
      })
      .catch(err => {
        console.error('Помилка отримання статистики:', err);
      });
```

4 ТЕСТУВАННЯ ВЕБ-ДОДАТКУ

4.1 User Interface Testing

Тестування роботи інтерфейсу користувача називається UI-тестування (User Interface Testing). Воно полягає в перевірці правильності функціонування елементів інтерфейсу користувача, таких як кнопки, форми, меню, а також перевіряє зовнішній вигляд і зручність використання.

На рис. 4.1 представлено головну сторінку.

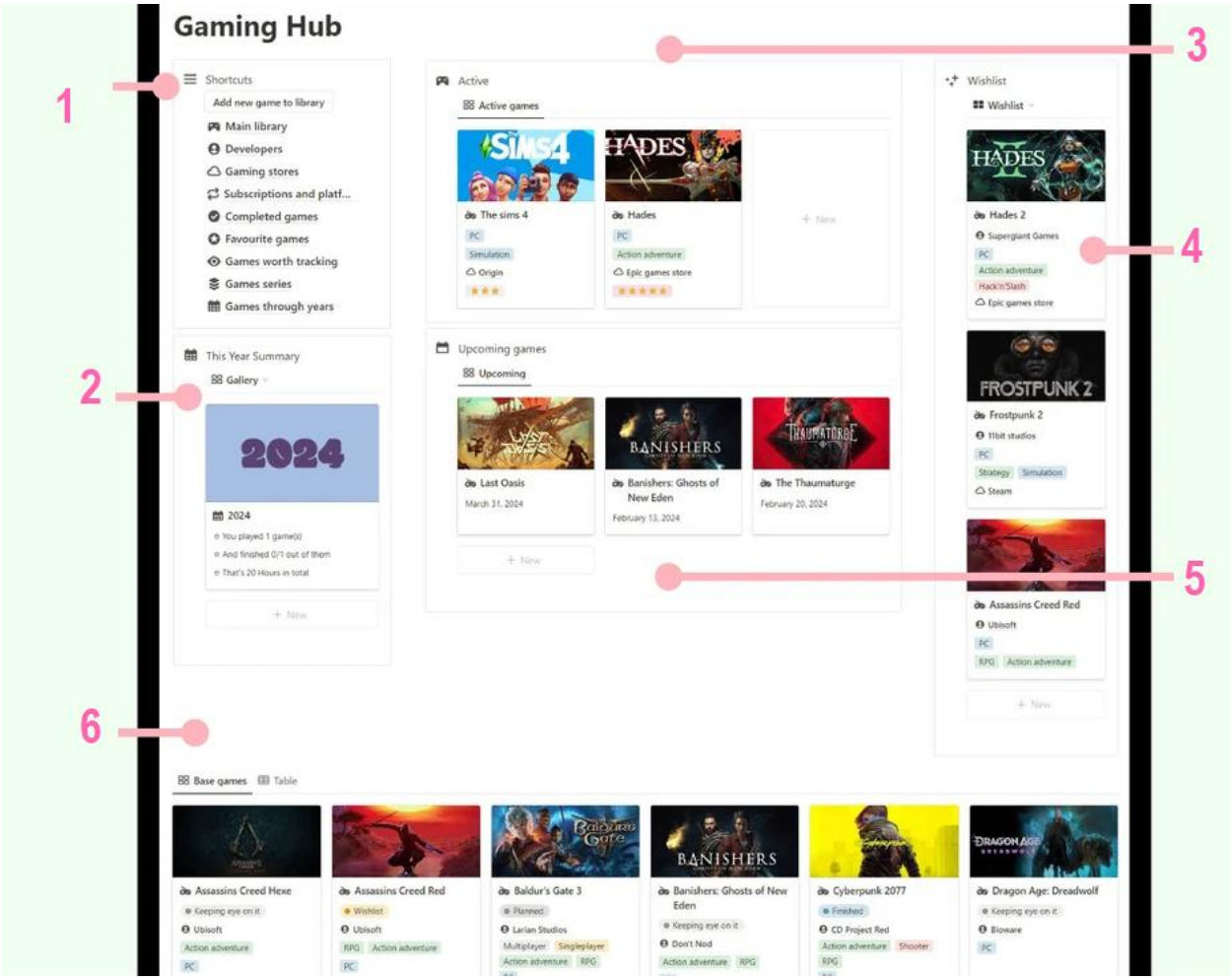

Рисунок 4.1 – Скріншот головної сторінки геймтрекеру

Слід перевірити функціональність інтерфейсу, оцінку зручності використання інтерфейсу, наскільки він інтуїтивний для користувача та перевірка роботи інтерфейсу на різних браузерах та операційних системах.

Головна сторінка геймтрекеру містить такі блоки:

1. Блок зручної навігації (рис. 4.2).

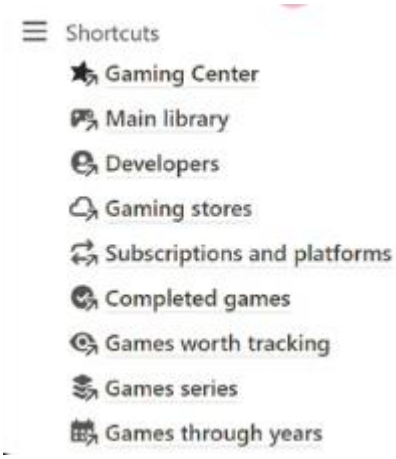

Рисунок 4.2 – Блок новігації

2. Блок результатів за рік.
3. Топ активних ігор (рис. 4.3).

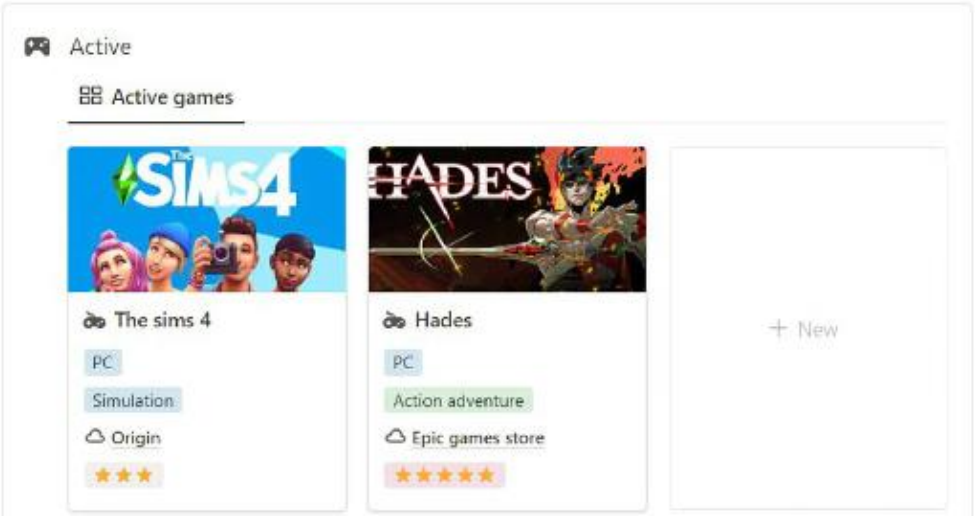

Рисунок 4.3 – Топ акттивних ігор

4. Список побажань користувача (рис. 4.4).
5. Анонс виходу нових ігор найближчим часом.
6. Швидкий доступ до основних ігор.

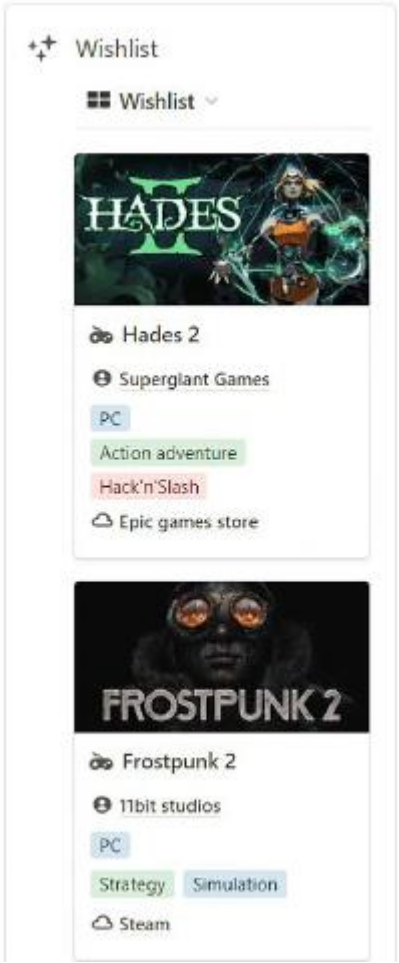

Рисунок 4.4 – Блок списку побажань користувача

При натискані на іконку гри користувачу відкривається вікно з переліком основної інформації (рис. 4.5). Ця інформація включає характеристику стану гри:

- «дії»;
- «стан»;
- «завершенність».

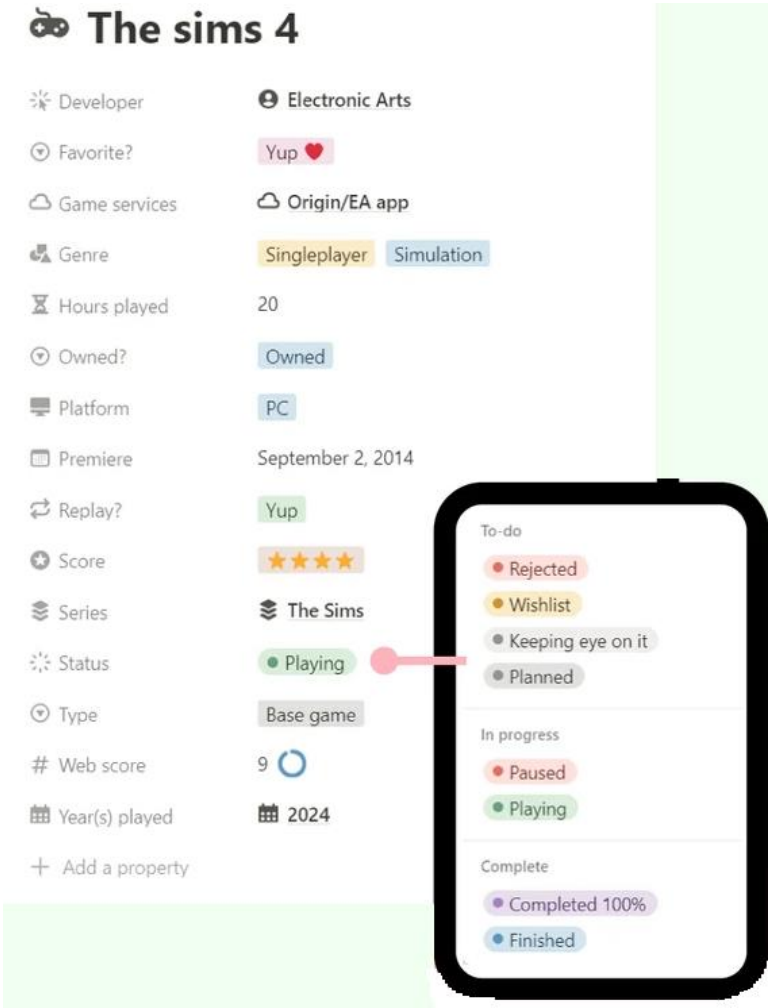

Рисунок 4.5 – Приклад вікна з повною інформацію щодо обраної гри

Приклад організації власної ігрової бібліотеки (рис. 4.6), де розміщено інформацію щодо жанрів ігор, балів гравця та інше. У цьому блоці реалізовано повну автоматизацію, тобто після оновлення основної бази інформація тут оновлюється. Також в цьому блоці розміщені сповіщення щодо найближчого виходу новинок, які схожі за жанром, вже існуючих у списку ігор користувача.

До кожної гри можна подивитись інформацію щодо її розробників та посилання на придбання ігрового контенту (рис. 4.7). Одна з головних вимог то геймтрекерів – це звіт за рік щодо успіхів користувача.

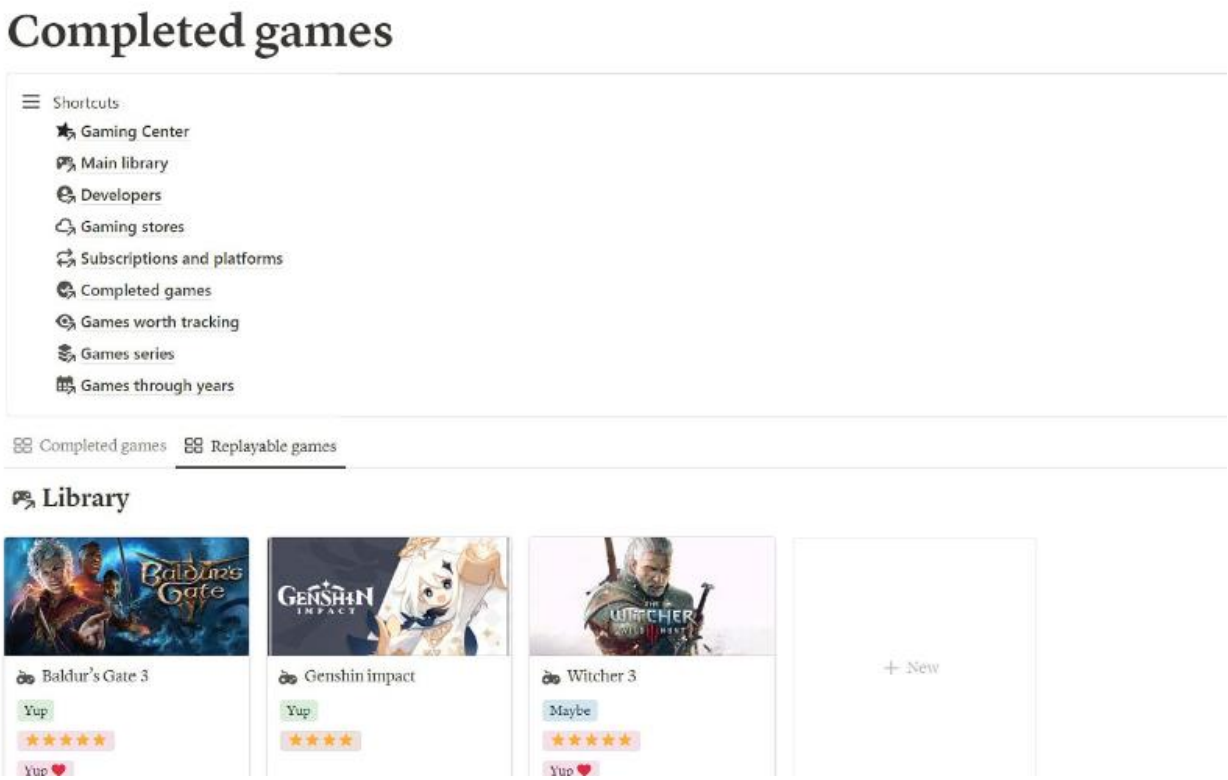

Рисунок 4.6 – Скріншот розділу «Library»

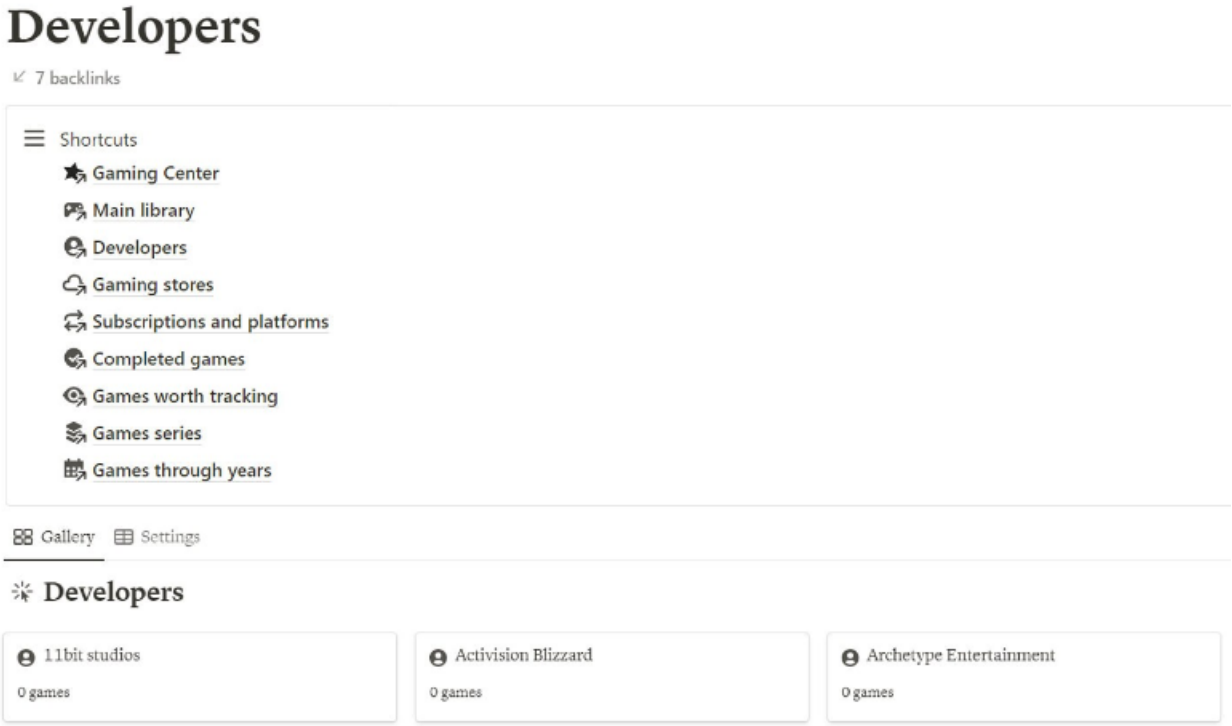

Рисунок 4.7 – Інформація щодо розробників ігрового контенту

Також, веб-додаток формує список рекомендацій відповідно до жанру ігор, які є у списку улюблений, та наявності популярних новинок на ринку (рис. 4.8). Приклад статистики за рік на рис. 4.9.

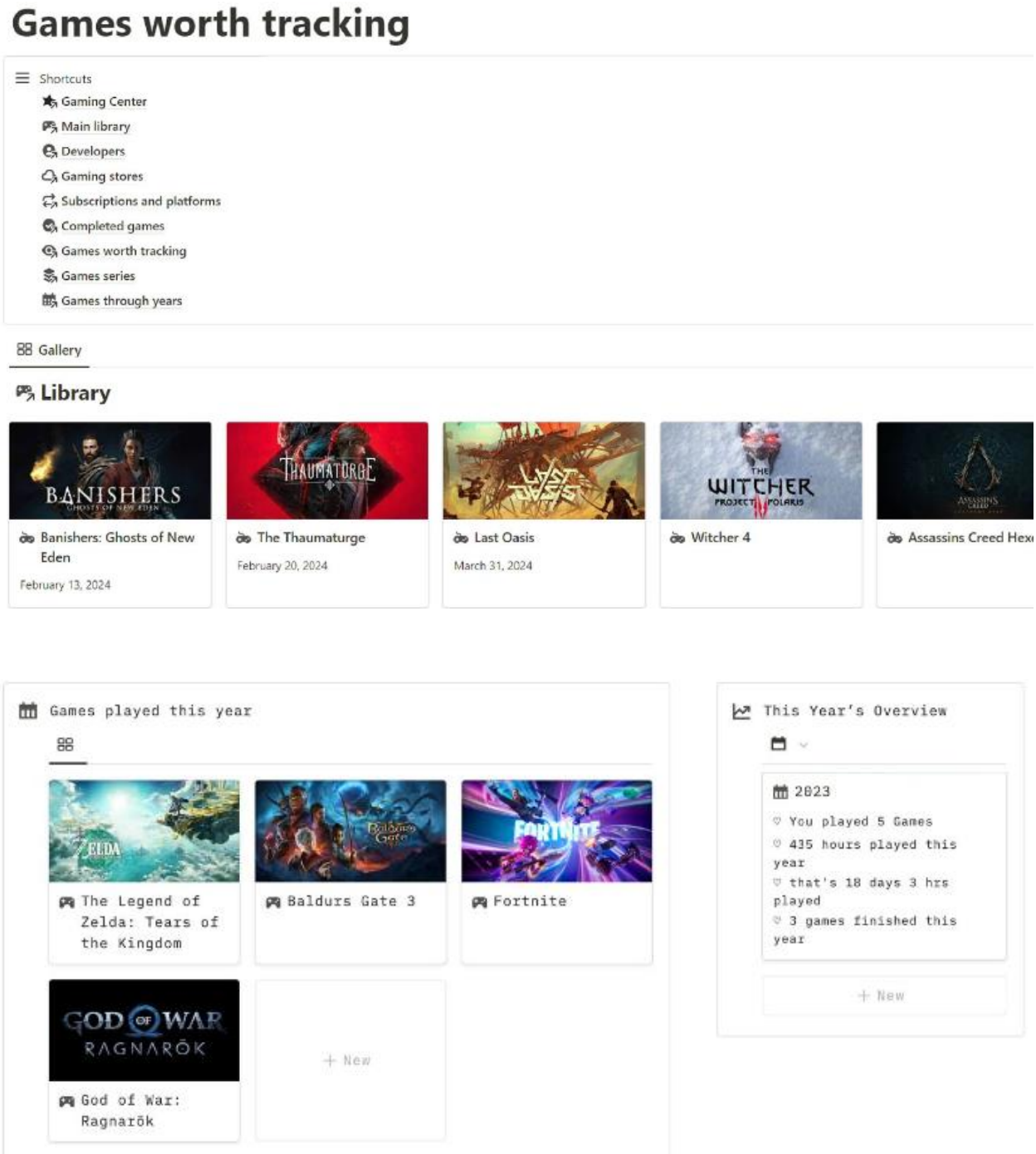

Рисунок 4.9 – Приклад статистики успіхів гравця за рік

4.2 Функціональне тестування веб-додатку

Тестування – це перевірка стану між реальною поведінкою програми та її очікуваною поведінкою на кінцевому наборі тестів, обраному певним чином. Тест – це спеціальна, штучно створена ситуація, обрана певним чином, і опис того, які спостереження за програмою потрібно зробити для перевірки її відповідності деяким вимогам.

Тесткейси – це документовані вказівки або скрипти, що описують послідовність дій для виконання певного тесту або набору тестів. Вони використовуються для створення, виконання та документування тестів програмного забезпечення.

Кожен тест-кейс повинен включати таки пункти, як:

1. Назва тест-кейсу: чітке визначення, що саме тестується.
2. Передумови: стан системи або умови, які повинні бути виконані до тесту.
3. Кроки: послідовність дій для виконання тесту.
4. Очікуваний результат.
5. Фактичний результат: результат, отриманий під час тестування.
6. Статус тесту: успішно/провалено.

Відповідно до варіантів використання, описаних вище, розроблені 8 тесткейсів (рис. 4.10-4.18).

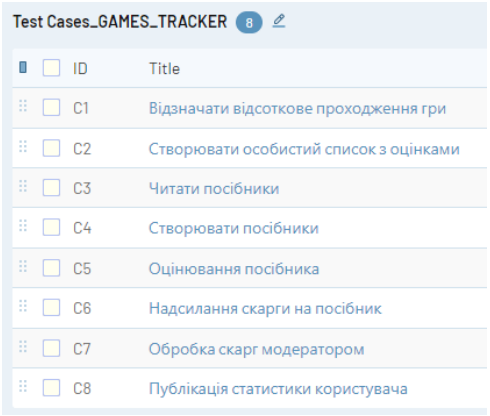

Рисунок 4.10 – Перелік тест-кейсів для функціонального тествування

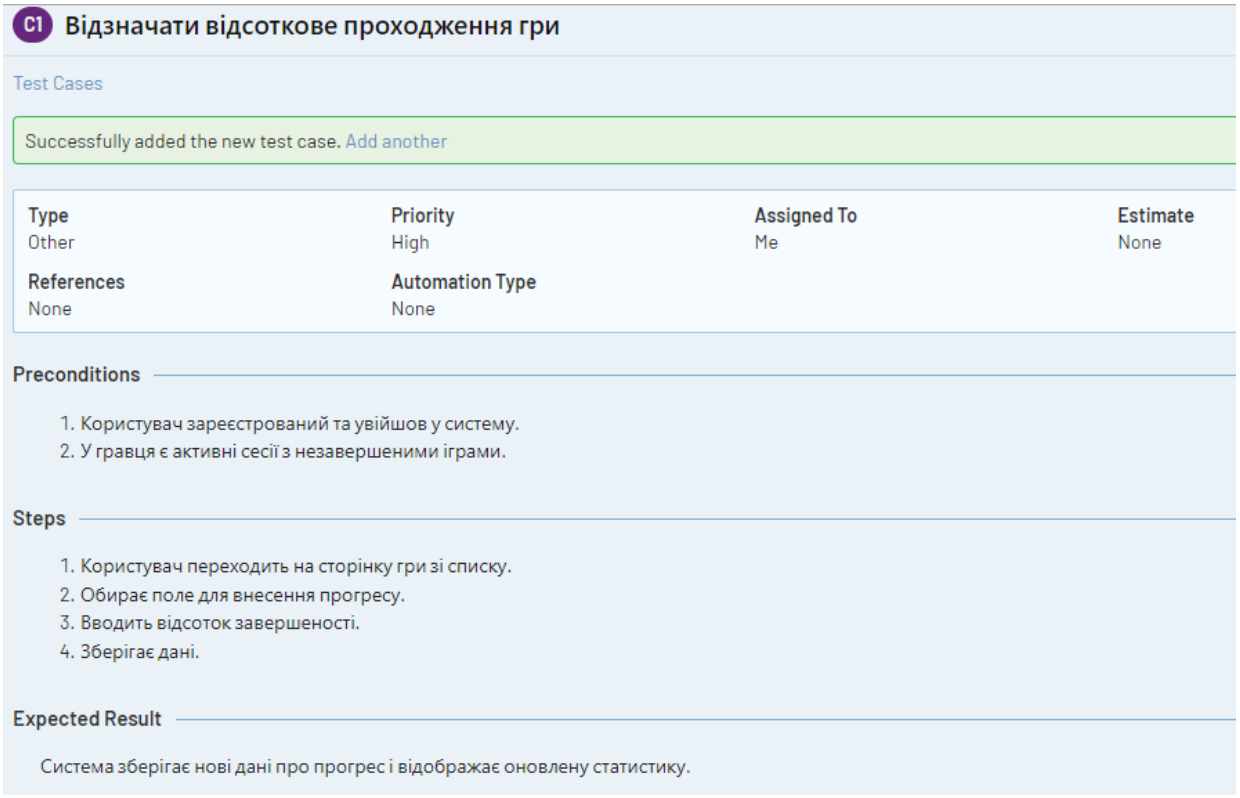

Рисунок 4.11 – Тест-кейс «Визначення відсоткового проходження гри»

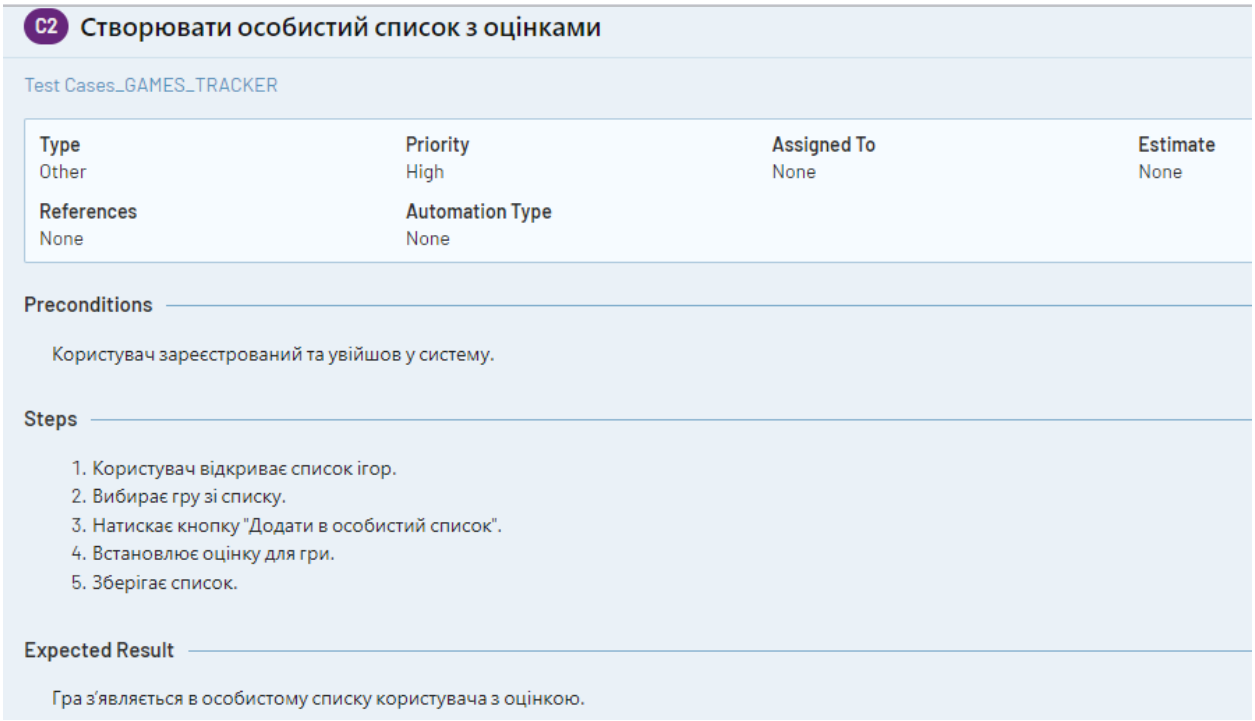

Рисунок 4.12 – Тест-кейс «Створення особистого спииску з оцінками»

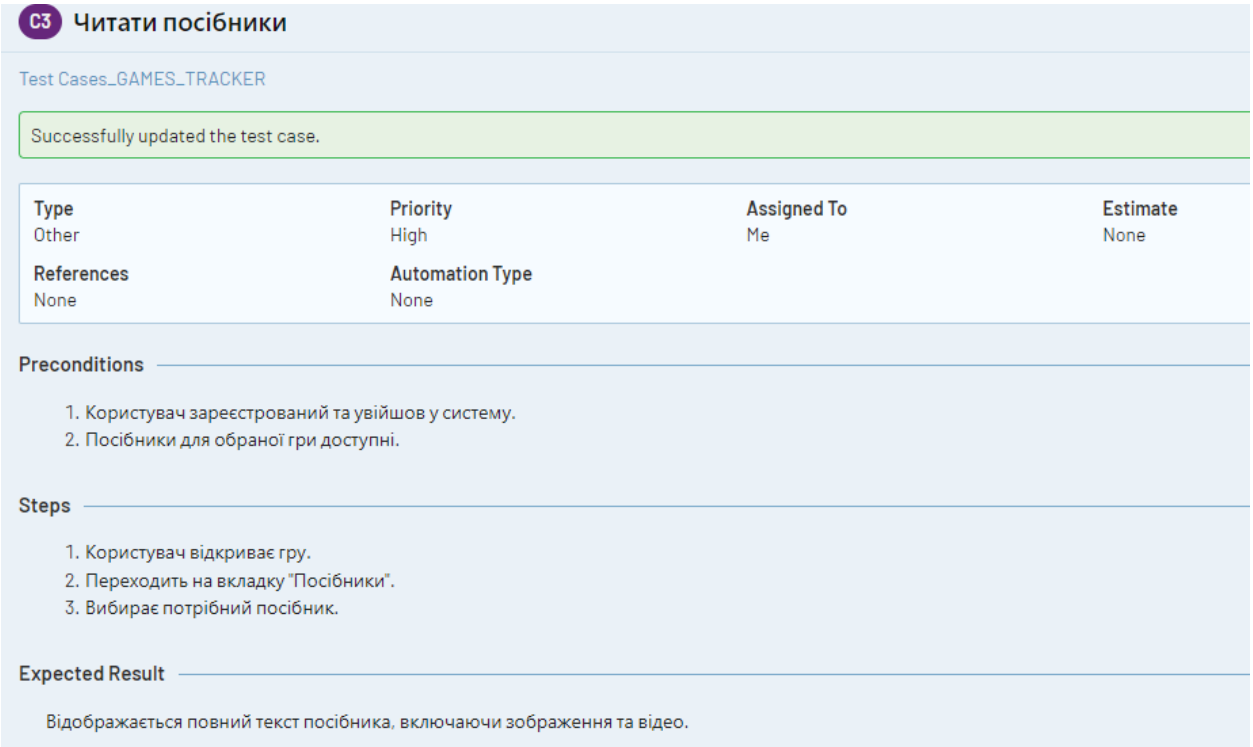

Рисунок 4.13 – Тест-кейс «Читання посібника»

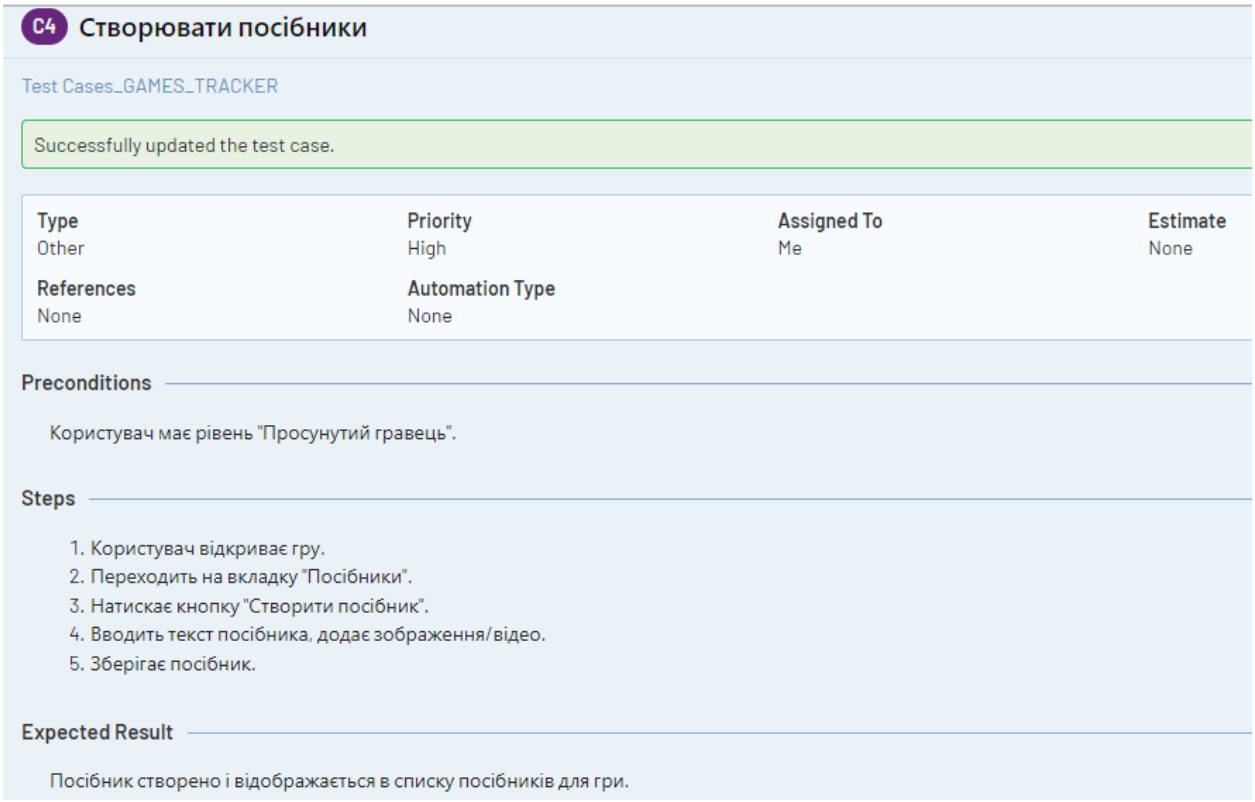

Рисунок 4.14 – Тест-кейс «Створення посібника»

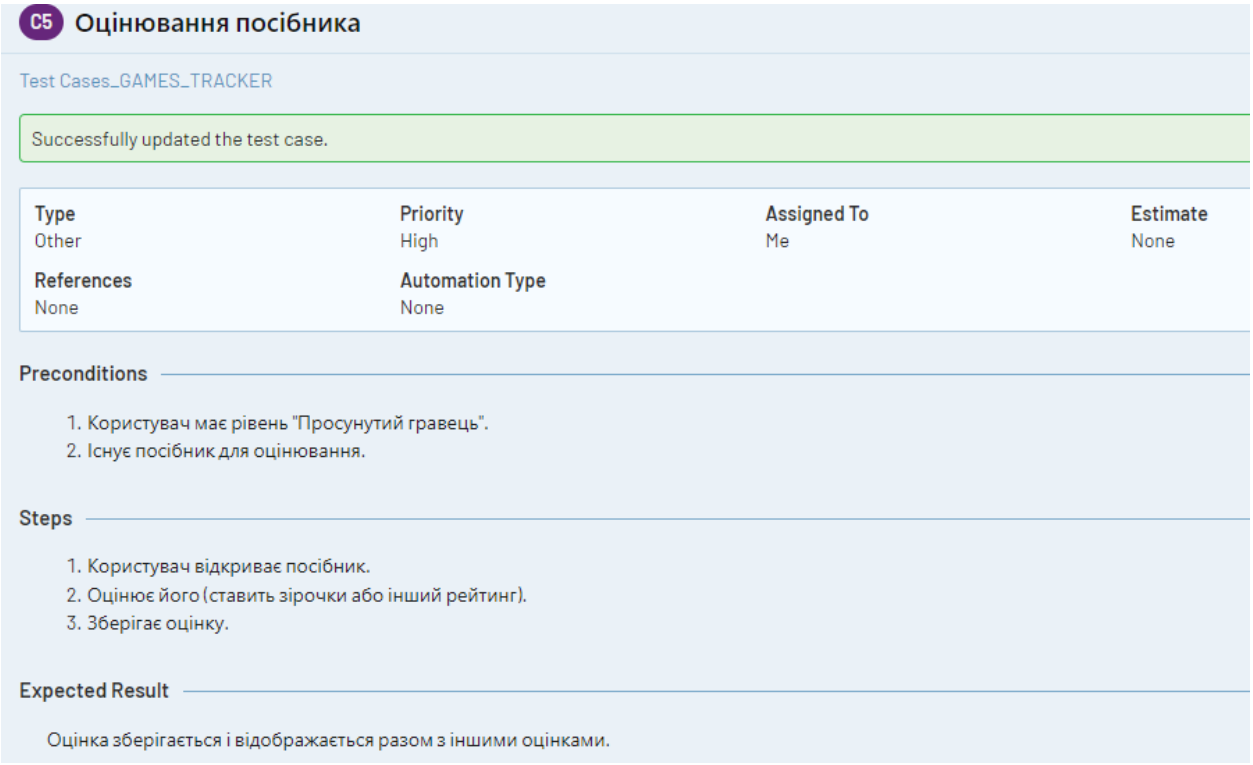

Рисунок 4.15 – Тест-кейс «Оцінка посібника»

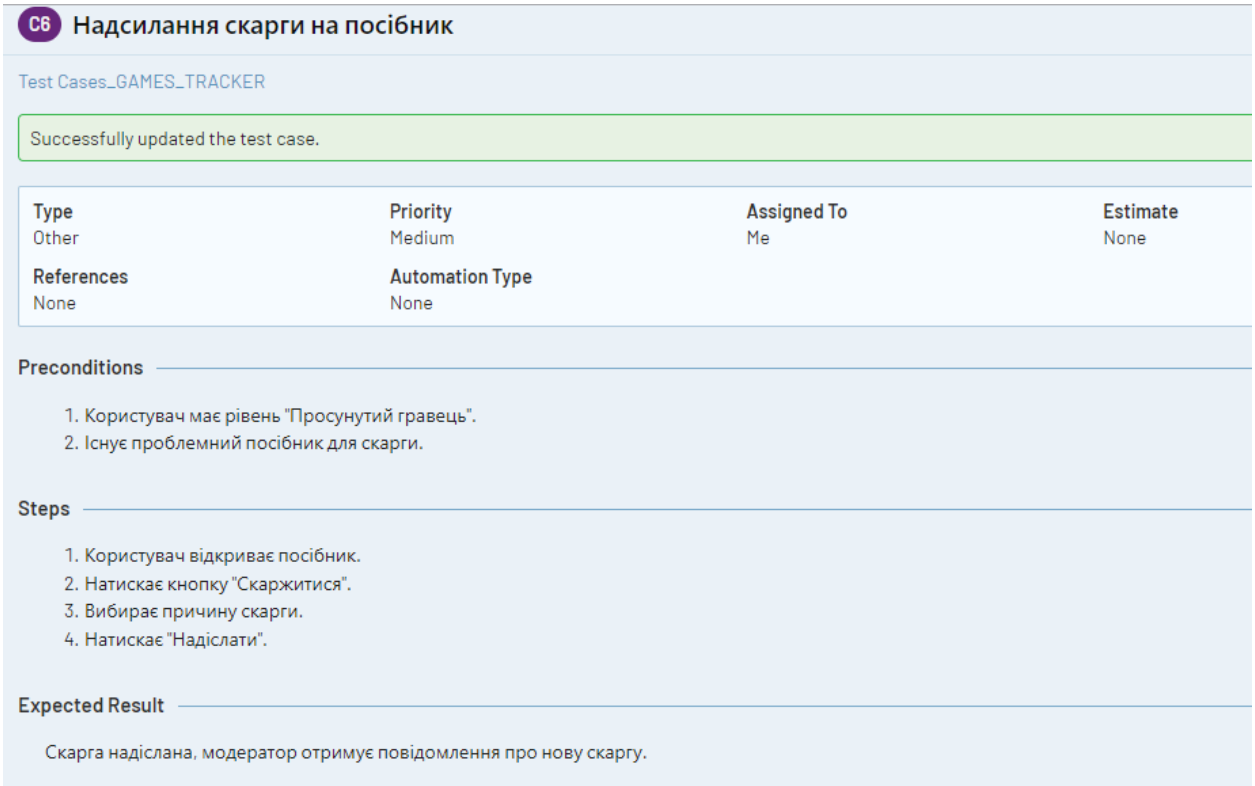

Рисунок 4.16 – Тест-кейс «Надсилання сарги на посібник»

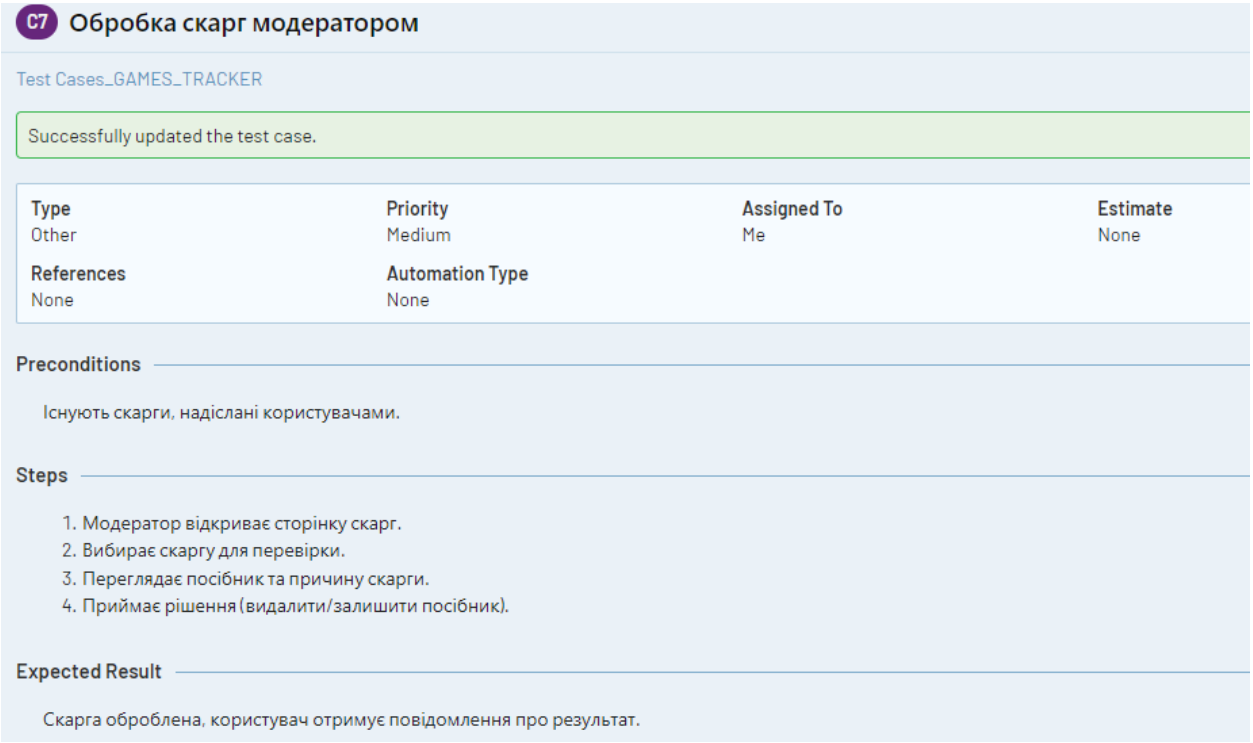

Рисунок 4.17 – Тест-кейс «Обробка скарг модератором»

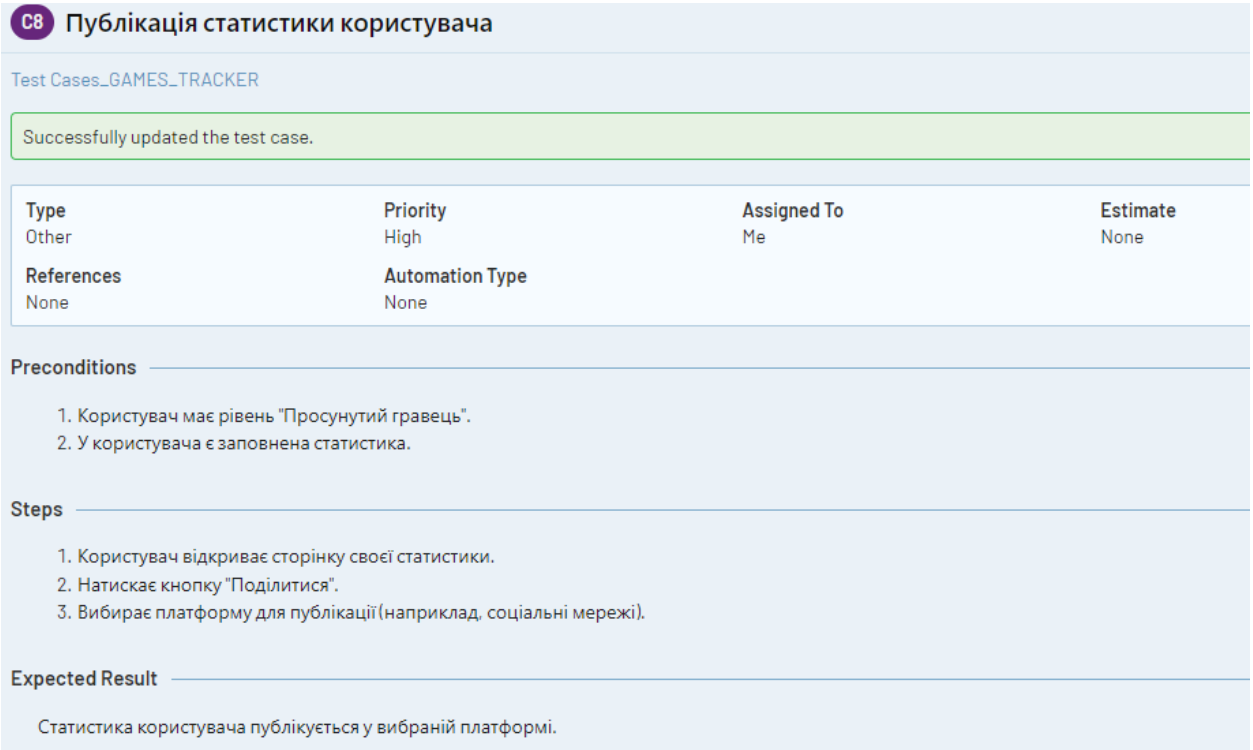

Рисунок 4.18 – Тест-кейс «Публікація статистики користувача»

Результати тестування представлено на рис. 4.19. Для всіх тест-кейсів підтверджено очікуваний результат.

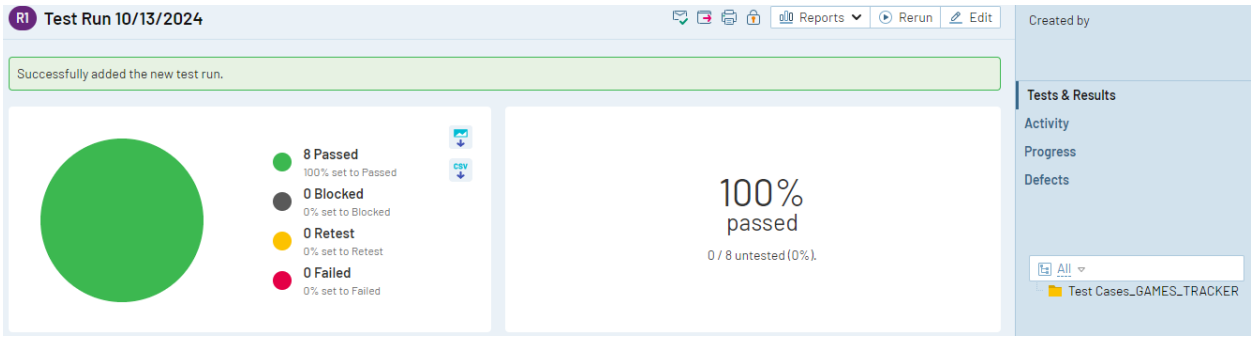

Рисунок 4.19 – Результати тестування

ВИСНОВКИ

Розробка геймтрекера є актуальним рішенням в умовах сучасного попиту на інструменти для моніторингу та аналізу прогресу гравців. Такий додаток вирішує проблему систематизації даних про досягнення, час гри, а також надає користувачам зручні функції для обміну досвідом та отримання рекомендацій.

Під час розробки геймтереку виконано системний аналіз, описані функціональні вимоги за побудована модель бізнес-процесів.

Було успішно розроблено та реалізовано архітектуру на основі MVC, що дозволяє легко масштабувати та адаптувати систему до нових вимог. Здійснено інтеграцію з платформою Steam, що надає додатку можливість автоматично отримувати та оновлювати дані про досягнення гравців. Створено структуру бази даних для збереження ключових даних про ігри, користувачів, прогрес і посібники.

Впроваджено функціонал для відстеження прогресу, створення посібників, перегляду статистики та отримання рекомендацій, що покращує загальний ігровий досвід. Реалізовано зручний та інтуїтивний інтерфейс на основі React, який забезпечує зручну навігацію по додатку та доступ до необхідної інформації.

Проведено функціональне тестування на основі тест-кейсів, результати якого свідчать про успішне досягнення мети розробки.

СПИСОК ВИКОРИСТОВАНИХ ДЖЕРЕЛ

1. Gametrack. URL: https://www.videogameseurope.eu/data-key-facts/gametrack-info/ (дата звернення 23.08.2024)
2. Tracker Network. URL: https://tracker.gg/ (дата звернення 23.08.2024)
3. The Best Video Game Tracker Apps (Like Goodreads for Video Games). URL: https://www.makeuseof.com/tag/like-goodreads-but-for-video-games-manage-your-game-collection-better/ (дата звернення 23.08.2024)
4. Gametrackers. URL: https://www.tripadvisor.com/Attraction_Review-g3442111-d12874555-Reviews-Gametrackers-Rongai_Rift_Valley_Province.html (дата звернення 23.08.2024)
5. GameTrack Review: An Elegant Way to Discover, Track, and Share Videogames. URL: https://www.macstories.net/reviews/gametrack-review-an-elegant-way-to-discover-track-and-share-videogames/ (дата звернення 23.08.2024)
6. Functions of the Game Tracker. URL: https://www.notion.so/templates/game-tracker (дата звернення 11.10.2024)
7. Why use React for web development in 2024: benefits, cases, top examples. URL: https://solveit.dev/blog/why-use-react-for-web-development (дата звернення 19.10.2024)
8. Build fast, responsive sites with Bootstrap. URL: https://getbootstrap.com/(дата звернення 19.10.2024)
9. Співбесіда з Node.js розробником. URL: https://dou.ua/lenta/articles/interview-node-js/ (дата звернення 19.10.2024)
10. Берген Р. Мова програмування JavaScript. ООО «Publishing House», 2021. – 162 с.
11. Сноткер В. JavaScript та Node.js. ООО «Publishing House», 2022. – 274 с.

12. Modern Software Testing Techniques: A Practical Guide for Developers and Testers 1st ed. Edition. Istvan Forgacs, Attila Kovacs: Apress, 2024.
13. Що таке Axios? URL: https://axios-http.com/uk/docs/intro (дата звернення 19.10.2024)