

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БІОРЕСУРСІВ І
ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій**

ПОГОДЖЕНО
Декан факультету

_____ **Ігор БОЛБОТ**
(підпис)

«___» _____ 2026 р.

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
Завідувач кафедри комп'ютерних наук

_____ **Белла ГОЛУБ**
(підпис)

«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

**на тему: «Програмне забезпечення з колекціонування та оцінювання відеоігор
за рейтингами користувачів»**

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Гарант освітньої програми

к.т.н., доцент

_____ **Ганна ВАЙГАНГ**
(підпис)

**Керівник бакалаврської
кваліфікаційної роботи**

ст.викладач

_____ **Віктор ПАНКРАТЬСВ**
(підпис)

Виконав

_____ **Станіслав САПЕГА**
(підпис)

Київ-2026

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

Факультет інформаційних технологій

**ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук**

К.Т.Н., доцент _____ Белла ГОЛУБ
(підпис)

«___» _____ 2025 р.

**ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ**

Сапезі Станіславу Олександровичу
(прізвище, ім'я, по батькові)

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Тема бакалаврської кваліфікаційної роботи

«Програмне забезпечення з колекціонування та оцінювання відеоігор за рейтингами користувачів»»

затверджена наказом від «19.12.2025» № 3204«С»

Термін подання завершеної роботи на кафедру «22» травня 2026 р.

Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту):

Перелік питань, що підлягають дослідженню:

Перелік графічних документів (за необхідності).

Дата видачі завдання «19» грудня 2025р.

**Керівник бакалаврської
кваліфікаційної роботи**

_____ (підпис)

Віктор ПАНКРАТ'ЄВ

**Завдання взяв до
виконання**

_____ (підпис)

Станіслав САПЕГА

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	5
ВСТУП.....	6
1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	9
1.1 Опис предметної області.....	9
1.2 Аналіз існуючих рішень.....	12
1.3 Моделювання програмної системи.....	17
1.4 Аналіз вимог інформаційної системи.....	22
1.5 Постановка завдання	27
2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	30
2.1 Логічна модель даних у вигляді ER-діаграми.....	30
2.2 Діаграма класів і кооперації	33
2.3 Діаграма компонентів розроблювальної системи	37
2.4 Діаграма пакетів	39
2.5 Висновки до другого розділу	41
3 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ ТА РОЗРОБЛЕННЯ АЛГОРИТМІЧНОГО ЗАБЕЗПЕЧЕННЯ.....	44
3.1 Обґрунтування вибору технологічних засобів та інструментів реалізації.....	44
3.2 Фізична модель даних розроблюваної системи.....	46
3.3 Алгоритмізація програмних модулів системи.....	49
3.4 Висновки до третього розділу	55

4 ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ ...	58
4.1 Тестування програмних модулів системи колекціонування та оцінювання відеоігор.....	58
4.2 Тестування інтерфейсних модулів системи колекціонування та оцінювання відеоігор.....	62
4.3 Результати тестування та аналіз ефективності системи, склад інсталяційного пакету	68
4.4 Висновки до четвертого розділу	72
ВИСНОВКИ	74
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	77
ДОДАТОК А	79
ДОДАТОК Б.....	86
ДОДАТОК В	89
ДОДАТОК Г.....	96

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

1. API – Application Programming Interface, програмний інтерфейс прикладного програмування
2. BPMN – Business Process Model and Notation, нотація моделювання бізнес-процесів
3. CSV – Comma-Separated Values, текстовий формат подання табличних даних
4. CRUD – Create, Read, Update, Delete, базові операції створення, перегляду, редагування та видалення даних
5. DB – Database, база даних
6. DFD – Data Flow Diagram, діаграма потоків даних
7. ER – Entity-Relationship, модель «сутність–зв’язок»
8. FK – Foreign Key, зовнішній ключ
9. GUI – Graphical User Interface, графічний інтерфейс користувача
10. PK – Primary Key, первинний ключ
11. QSS – Qt Style Sheets, механізм стилізації інтерфейсу в Qt-застосунках
12. RBAC – Role-Based Access Control, рольова модель керування доступом
13. SQL – Structured Query Language, мова структурованих запитів
14. SQLite – вбудована реляційна система керування базами даних
15. UML – Unified Modeling Language, уніфікована мова моделювання
16. БД – база даних
17. ЗВО – заклад вищої освіти
18. ІС – інформаційна система
19. ОС – операційна система
20. ПЗ – програмне забезпечення
21. СКБД – система керування базами даних
22. UI – User Interface, користувацький інтерфейс

ВСТУП

Актуальність теми кваліфікаційної роботи зумовлена активним розвитком індустрії відеоігор, зростанням кількості цифрових і фізичних ігрових копій у користувачів, а також потребою у зручних програмних засобах для систематизації персональних ігрових колекцій. Сучасний користувач взаємодіє з великою кількістю ігрових платформ, онлайн-магазинів, сервісів цифрової дистрибуції та спільнот, унаслідок чого інформація про придбані ігри, проходження, оцінки, відгуки та особисті вподобання часто зберігається фрагментарно. Це ускладнює контроль власної колекції, аналіз ігрової активності, пошук потрібної гри та формування об'єктивного рейтингу на основі користувацьких оцінок.

Процес колекціонування відеоігор охоплює комплекс взаємопов'язаних дій, зокрема додавання гри до особистої бібліотеки, зазначення типу копії, фіксацію платформи, жанру, розробника, видавця, статусу проходження, кількості зіграних годин, а також виставлення оцінки й написання відгуку. За відсутності спеціалізованого програмного забезпечення користувачі часто застосовують таблиці, нотатки або окремі неузгоджені сервіси, що не забезпечують цілісного обліку ігрової колекції та не дозволяють зручно поєднати персональні записи з рейтинговою системою.

Традиційні підходи до ведення ігрових списків здебільшого орієнтовані або на простий перелік ігор, або на загальний каталог без повної підтримки індивідуальної колекції користувача. У таких рішеннях не завжди передбачено розмежування цифрових і фізичних копій, гнучкий облік статусів проходження, збереження відгуків, аналіз користувацьких рейтингів та централізовану роботу з довідниками жанрів, платформ, розробників і видавців. Це знижує зручність користування системою та обмежує можливості подальшого аналізу накопичених даних.

Використання сучасних веб-технологій і клієнт–серверної архітектури відкриває можливості для створення інформаційної системи, яка забезпечує централізоване зберігання даних про відеоігри, користувачів, колекції, оцінки та відгуки. Застосування серверного API, реляційної бази даних і модульної структури програмного забезпечення дозволяє підвищити надійність, масштабованість і зручність підтримки системи. Такий підхід забезпечує ефективну взаємодію між клієнтським інтерфейсом, серверною логікою та базою даних, а також створює основу для подальшого розширення функціоналу.

Метою кваліфікаційної роботи є проектування та розробка програмного забезпечення з колекціонування та оцінювання відеоігор за рейтингом користувачів, що забезпечує облік ігрових колекцій, ведення каталогу відеоігор, збереження оцінок і відгуків, а також формування рейтингової інформації на основі користувацьких даних.

Для досягнення поставленої мети в роботі передбачається розв’язання таких **завдань**:

1. дослідити предметну область колекціонування відеоігор, користувацького оцінювання та ведення ігрових каталогів;
2. проаналізувати існуючі програмні рішення для обліку відеоігор, бібліотек користувачів і рейтингових систем;
3. визначити функціональні, нефункціональні та безпекові вимоги до розроблюваної системи;
4. спроектувати логічну та фізичну модель бази даних для зберігання відомостей про користувачів, ігри, жанри, платформи, колекції, оцінки й відгуки;
5. розробити UML-моделі, що відображають структуру, компоненти та основні процеси роботи інформаційної системи;
6. реалізувати програмний прототип системи на основі клієнт–серверної архітектури;
7. провести тестування програмних модулів і оцінити ефективність роботи розробленої системи.

Об'єктом дослідження є процес колекціонування, систематизації та оцінювання відеоігор користувачами.

Предметом дослідження є методи, моделі та програмні засоби створення інформаційної системи для обліку ігрових колекцій, ведення каталогу відеоігор і формування рейтингу на основі користувацьких оцінок.

Наукова новизна роботи полягає у розробленні підходу до побудови інформаційної системи, яка поєднує функції персонального обліку відеоігор, класифікації ігрового контенту, збереження користувацьких оцінок і формування рейтингової інформації в межах єдиного програмного середовища. Запропоноване рішення передбачає розмежування цифрових і фізичних копій ігор, підтримку статусів проходження, облік ігрового часу, ведення відгуків і використання структурованої бази даних для подальшого аналізу інформації.

Методи дослідження включають системний аналіз предметної області, об'єктно-орієнтоване проєктування, UML-моделювання, ER-моделювання, нормалізацію структури бази даних, а також методи програмної реалізації клієнт–серверних інформаційних систем. Для організації взаємодії між компонентами системи використовується REST-підхід, серверна частина може бути реалізована на основі Python API, а для збереження даних застосовується реляційна база даних PostgreSQL.

Практична значущість одержаних результатів полягає у можливості використання розробленої системи для ведення особистих ігрових колекцій, обліку цифрових і фізичних копій відеоігор, фіксації статусу проходження, збереження оцінок і відгуків користувачів. Система може бути корисною для геймерів, колекціонерів, адміністраторів ігрових каталогів та спільнот, які потребують зручного інструменту для організації інформації про відеоігри та формування рейтингу на основі реального користувацького досвіду.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Предметна область розроблюваної системи охоплює процеси ведення персональних колекцій відеоігор, систематизації ігрового контенту, збереження користувацьких оцінок і формування рейтингу відеоігор на основі відгуків користувачів. У сучасних умовах відеоігри поширюються через різні цифрові платформи, онлайн-магазини, консолі та фізичні носії, тому користувачеві складно централізовано контролювати власну бібліотеку, статус проходження, кількість зіграних годин, наявність фізичних або цифрових копій, а також особисті оцінки й відгуки.

Основна проблема предметної області полягає у фрагментарності зберігання інформації про відеоігри. Частина даних може міститися у цифрових сервісах, частина - у фізичній колекції користувача, а оцінки та відгуки часто залишаються на різних платформах або взагалі не систематизуються. Це ускладнює пошук потрібної гри, аналіз власних ігрових вподобань, перегляд історії проходження та формування об'єктивного користувацького рейтингу.

Розроблюване програмне забезпечення призначене для об'єднання зазначених процесів в одному інформаційному середовищі. Система має забезпечувати облік користувачів, ведення каталогу відеоігор, класифікацію ігор за жанрами та платформами, збереження інформації про розробників і видавців, формування персональних колекцій, фіксацію статусу проходження, додавання оцінок і відгуків, а також перегляд рейтингової та статистичної інформації.

У межах предметної області основними учасниками є користувач і адміністратор. Користувач працює з особистою колекцією, додає ігри, змінює їх статус, вказує кількість годин, залишає оцінки й відгуки. Адміністратор відповідає за підтримку каталогу відеоігор, актуальність довідників жанрів,

платформ, розробників і видавців, а також за контроль коректності даних у системі.

Загальна характеристика предметної області наведена в таблиці 1.1.

Таблиця 1.1

Характеристика предметної області системи колекціонування та оцінювання відеоігор

Елемент предметної області	Зміст	Основні дані та результати
Користувачі системи	Особи, які ведуть власну ігрову колекцію та оцінюють відеоігри	Обліковий запис, email, пароль, статус користувача, дата реєстрації
Каталог відеоігор	Централізований перелік відеоігор, доступних для додавання до колекції	Назва гри, рік випуску, опис, віковий рейтинг, розробник, видавець
Жанри та платформи	Довідкова інформація для класифікації відеоігор	Жанр, ігрова платформа, виробник платформи
Колекція користувача	Персональний набір ігор, доданих користувачем до власної бібліотеки	Гра, статус проходження, дата додавання, кількість годин, примітки
Тип копії гри	Поділ елементів колекції на цифрові та фізичні копії	Назва магазину, email акаунта, ліцензійний ключ, видання, стан носія, штрихкод
Оцінювання відеоігор	Фіксація користувацької оцінки та текстового відгуку	Оцінка від 1 до 10, текст відгуку, дата оцінювання
Формування рейтингу	Узагальнення оцінок користувачів для визначення популярності гри	Середній бал, кількість оцінок, список найкращих ігор
Адміністрування	Підтримка коректності каталогу та довідкових даних	Додавання, редагування та оновлення даних про ігри, жанри, платформи
Аналітична інформація	Надання користувачам узагальнених даних про ігри та колекції	Статистика проходження, рейтинги, кількість ігор у колекції, популярні жанри

Основними інформаційними об'єктами системи є користувач, відеогра, розробник, видавець, жанр, платформа, елемент колекції, цифрова копія, фізична копія, оцінка та відгук. Зв'язок між користувачем і відеоігрою реалізується через елемент колекції, оскільки одна й та сама гра може бути додана різними користувачами з різними статусами проходження, кількістю годин, примітками

та типом копії. Оцінка й відгук також прив'язуються до конкретного користувача та конкретної гри, що дає змогу формувати рейтинг на основі реального користувацького досвіду. DFD-0го рівня діаграма представлена на рисунку 1.1

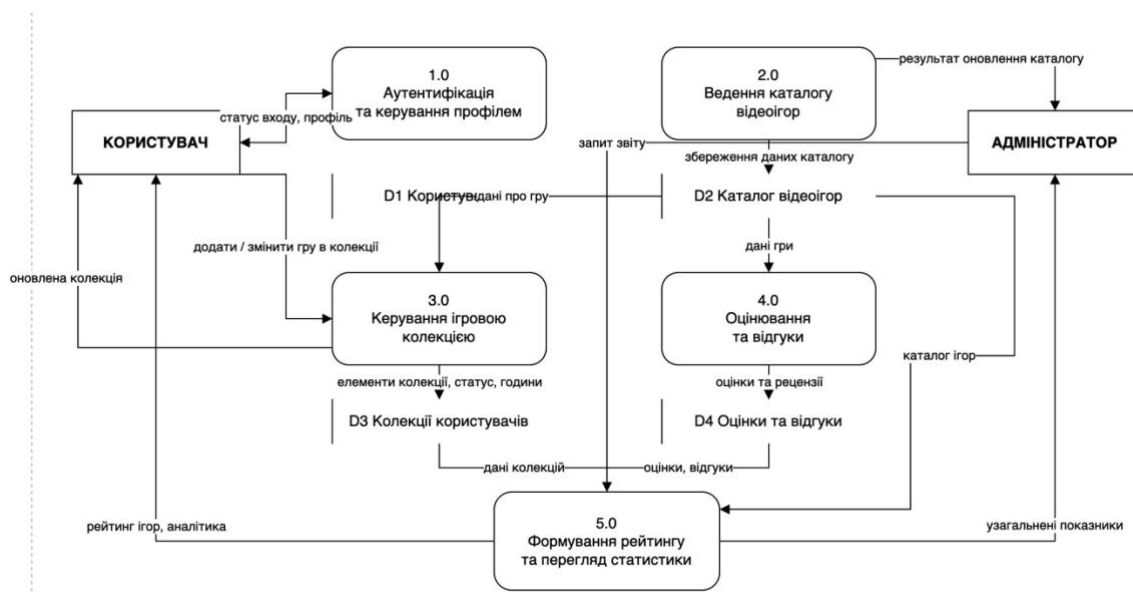


Рисунок 1.1 – DFD-0го рівня системи колекціонування відеоігор

Для системи важливо підтримувати розмежування цифрових і фізичних копій. Цифрова копія може містити інформацію про магазин, акаунт або ліцензійний ключ, тоді як фізична копія описується виданням, станом носія та штрихкодом. Такий підхід дає змогу використовувати систему не лише як простий список ігор, а як повноцінний інструмент для колекціонерів.

Окреме значення має механізм оцінювання. Користувач може виставити грі бал, залишити текстовий відгук, а система на основі таких даних формує середній рейтинг. Це дозволяє не лише зберігати персональні враження, а й створювати загальну рейтингову інформацію, корисну для інших користувачів під час вибору гри.

У межах інформаційної системи доцільно виділити такі основні процеси: реєстрація та вхід користувача, перегляд каталогу відеоігор, додавання гри до колекції, редагування статусу проходження, внесення оцінки та відгуку, перегляд рейтингу, адміністрування довідників і формування статистичної інформації. Усі ці процеси мають бути реалізовані в єдиному програмному середовищі з централізованою базою даних

1.2 Аналіз існуючих рішень

Для обґрунтування доцільності розроблення програмного забезпечення з колекціонування та оцінювання відеоігор проведено аналіз наявних сервісів, які частково вирішують подібні завдання. Основну увагу приділено системам, що забезпечують ведення ігрової бібліотеки, фіксацію статусу проходження, оцінювання ігор, написання відгуків, імпорт даних з інших платформ та роботу з каталогом відеоігор.

Серед найбільш поширених рішень у цій предметній області можна виділити Backloggd, Grouvee, Playnite, RAWG та IGDB. Вони реалізують різні підходи: одні орієнтовані на персональний беклог і соціальну взаємодію, інші — на локальне об'єднання ігрових бібліотек або надання довідкових даних про відеоігри через базу даних чи API.

На рис. 1.2 подано інтерфейс сервісу Backloggd. Ця система призначена для відстеження ігрової колекції, ведення беклогу, оцінювання пройдених ігор, додавання ігор до списку бажаного та перегляду активності інших користувачів. Перевагою Backloggd є зручна організація персональної бібліотеки та наявність соціальних функцій. Водночас сервіс не орієнтований на детальне розмежування фізичних і цифрових копій гри, а також не забезпечує повноцінного адміністрування власної структури каталогу в межах окремої інформаційної системи.

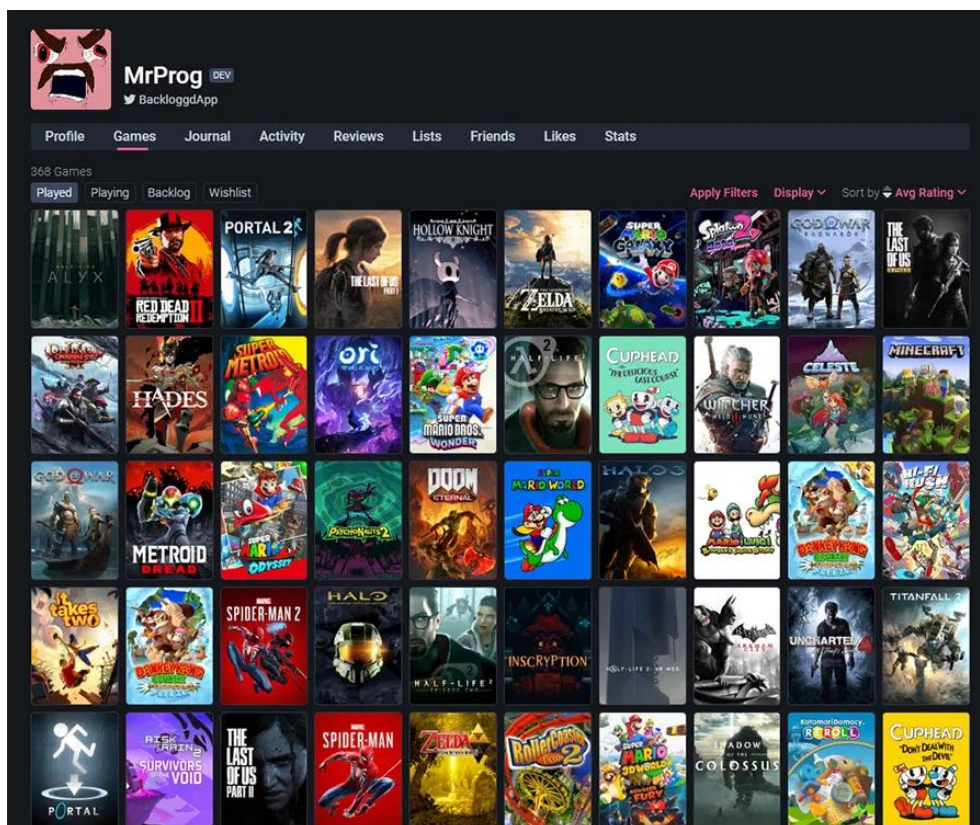


Рисунок 1.2 – Інтерфейс сервісу Backlogg'd для ведення ігрової колекції

На рис. 1.3 наведено приклад інтерфейсу Grouvee. Система дозволяє каталогізувати ігрову колекцію, вести список ігор у статусах Playing, Played, Backlog, Wish List, створювати власні полиці, оцінювати й рецензувати ігри, імпортувати бібліотеку Steam та переглядати активність друзів. Основним недоліком Grouvee у контексті розроблюваної системи є залежність від зовнішнього сервісу та обмежені можливості адаптації структури даних під власні вимоги.

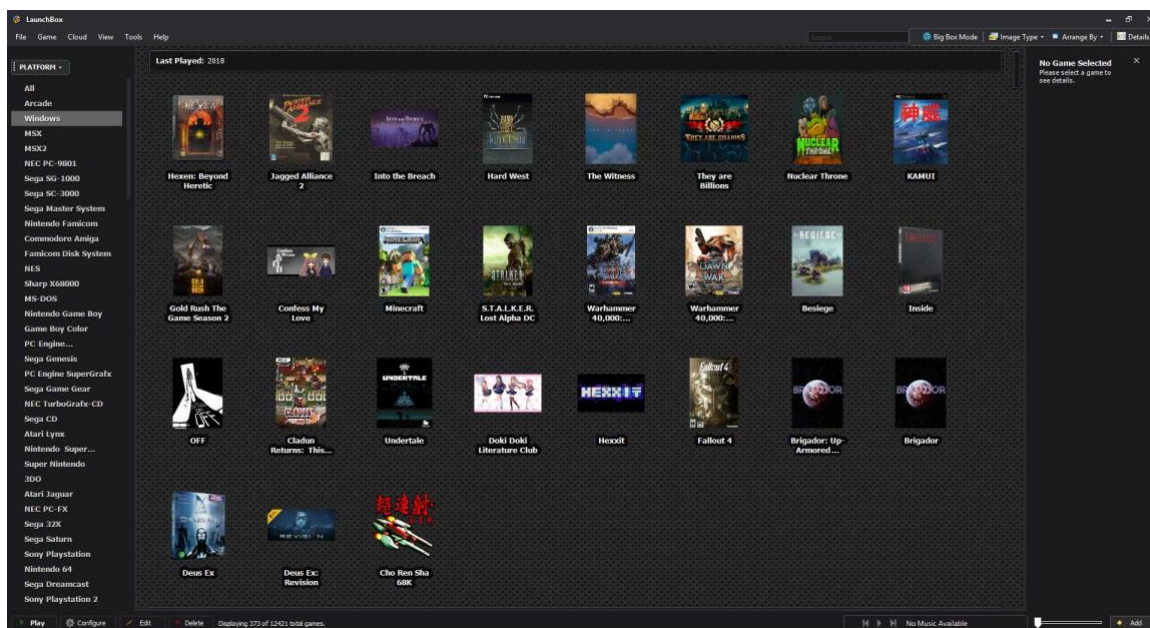


Рисунок 1.3 – Інтерфейс сервісу Grouvee для обліку беклогу та оцінювання ігор

На рис. 1.4 представлено Playnite - локальний менеджер ігрових бібліотек. Його перевагою є можливість об'єднання ігор із різних платформ в одному інтерфейсі, підтримка сторонніх бібліотек, емуляторів, розширень і автоматичне завантаження метаданих, зокрема обкладинок, описів та іншої інформації про ігри. Однак Playnite більше орієнтований на запуск ігор і керування локальною бібліотекою, ніж на формування спільного рейтингу, написання відгуків і роботу з користувацькими оцінками у вебсистемі.

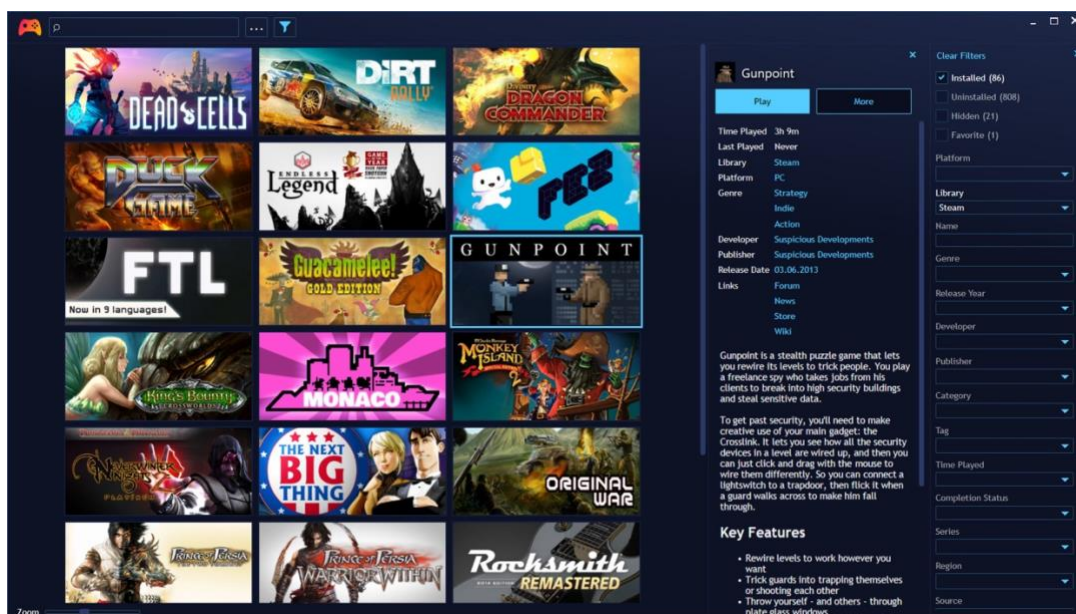


Рисунок 1.4 – Інтерфейс Playnite для об'єднання ігрових бібліотек

На рис. 1.5 зображено сервіс RAWG, який поєднує функції бази даних відеоігор, рейтингової платформи та джерела метаданих. RAWG містить дані про ігри, платформи, жанри, розробників, видавців, оцінки, середній час проходження, скріншоти, трейлери та посилання на магазини. Перевагою рішення є велика база даних і наявність API, проте система не призначена для розгортання як власний керований програмний продукт із повністю контрольованою структурою бази даних і бізнес-логікою.

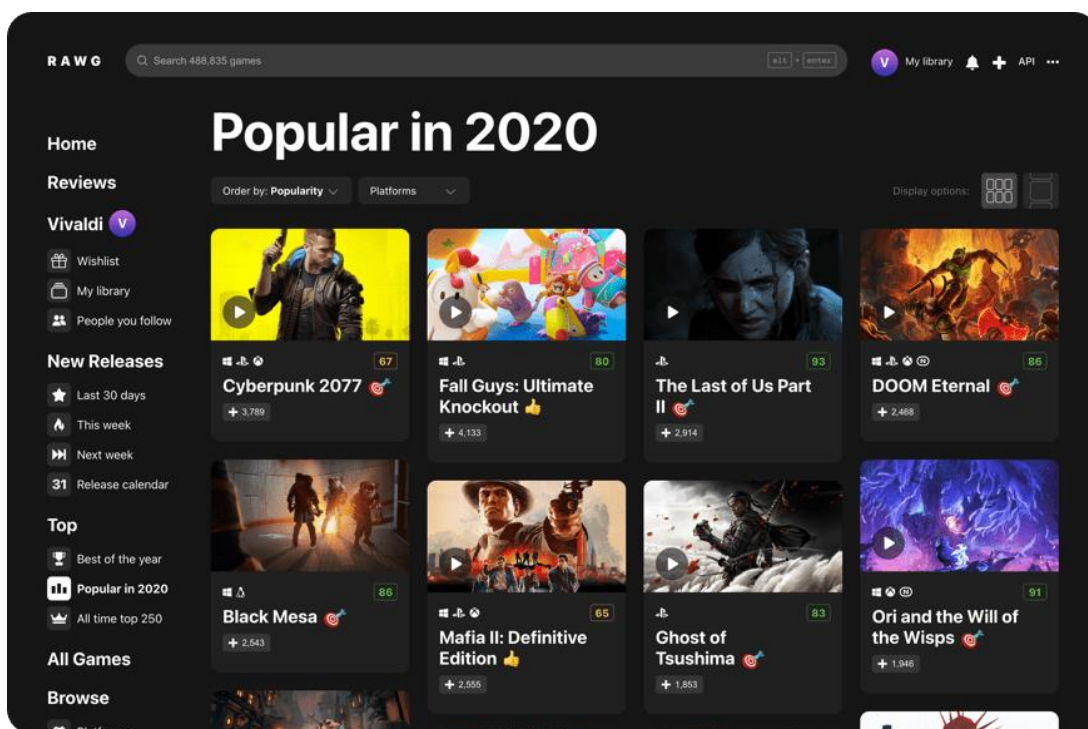


Рисунок 1.5 – Інтерфейс RAWG як бази даних відеоігор

На рис. 1.6 подано приклад роботи з IGDB. Цей сервіс є базою даних відеоігор і надає API для отримання інформації про ігри, зображення, відео та інші метадані. IGDB може використовуватися як джерело довідкової інформації про ігри, однак самостійно не вирішує задачу ведення персональної колекції користувача з поділом на цифрові й фізичні копії, статуси проходження, власні нотатки та локальне адміністрування системи.

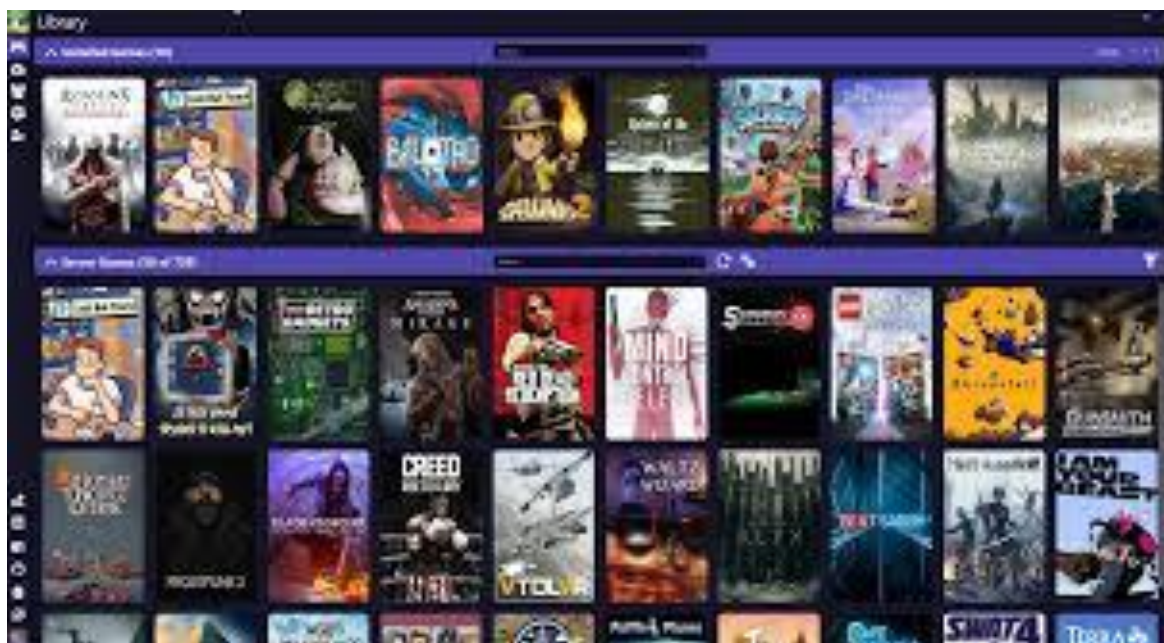


Рисунок 1.6 – Інтерфейс IGDB як джерела довідкової інформації про відеоігри

Порівняння існуючих рішень із розроблюваною системою наведено в таблиці 1.2.

Таблиця 1.2

Порівняння існуючих рішень та розроблюваної системи

Система	Основне призначення	Колекція користувача	Оцінки та відгуки	Поділ на цифрові/фізичні копії	Основне обмеження
Backloggd	Ведення беклогу та ігрового профілю	Так	Так	Обмежено	Залежність від зовнішнього сервісу
Grouvee	Каталогізація ігор, полиці, рейтинги	Так	Так	Обмежено	Обмежена адаптація під власну БД
Playnite	Локальний менеджер ігрових бібліотек	Так	Обмежено	Частково	Немає повноцінної вебрейтингової системи
RAWG	База даних відеоігор та API	Частково	Так	Ні	Орієнтація на довідкові дані, а не власну систему

IGDB	Джерело метаданих про відеоігри	Частково	Частково	Ні	Переважно API та база даних
------	---------------------------------	----------	----------	----	-----------------------------

Результати аналізу показують, що існуючі рішення забезпечують лише окремі аспекти роботи з відеоіграми: ведення беклогу, перегляд каталогу, оцінювання, імпорт бібліотек або отримання метаданих. Проте вони не поєднують у повному обсязі персональне колекціонування, поділ копій на цифрові й фізичні, збереження статусів проходження, користувацькі відгуки, рейтинговий механізм і можливість адміністрування власної бази даних у межах однієї системи.

Отже, розроблення власного програмного забезпечення є доцільним. Запропонована система має забезпечити централізований облік відеоігор, керування колекціями користувачів, підтримку оцінок і відгуків, формування рейтингу та збереження структурованих даних у власній базі даних. Це дозволить усунути обмеження наявних сервісів і створити гнучке рішення, адаптоване до вимог кваліфікаційної роботи.

1.3 Моделювання програмної системи

Моделювання предметної області виконано з метою формалізації основних учасників, функцій і сценаріїв роботи програмного забезпечення з колекціонування та оцінювання відеоігор. Для опису системи використано UML-діаграми прецедентів, послідовності та активності. Такий підхід дозволяє визначити межі системи, розподіл ролей користувачів, логіку оброблення запитів та послідовність виконання основних операцій.

Розроблювана система орієнтована на підтримку трьох основних груп користувачів: геймера, модератора та адміністратора. Геймер працює з каталогом ігор, формує власну колекцію, додає ігри до списку бажаного, змінює статус проходження, залишає оцінки й відгуки. Модератор контролює якість користувацького контенту, перевіряє скарги, приховує або видаляє некоректні відгуки. Адміністратор відповідає за підтримку каталогу ігор, оновлення

довідкових даних, контроль структури системи та перегляд узагальненої аналітичної інформації.

На рисунку 1.7 подано діаграму прецедентів системи колекціонування та оцінювання відеоігор. Вона відображає основні функції, доступні кожному типу користувача. Центральною частиною моделі є межа системи, у якій розміщено функціональні можливості програмного забезпечення. Геймер взаємодіє з більшістю користувацьких функцій, оскільки саме він є основним джерелом даних про колекції, оцінки та відгуки. Модератор працює з процесами перевірки відгуків і скарг, а адміністратор виконує керування даними каталогу.

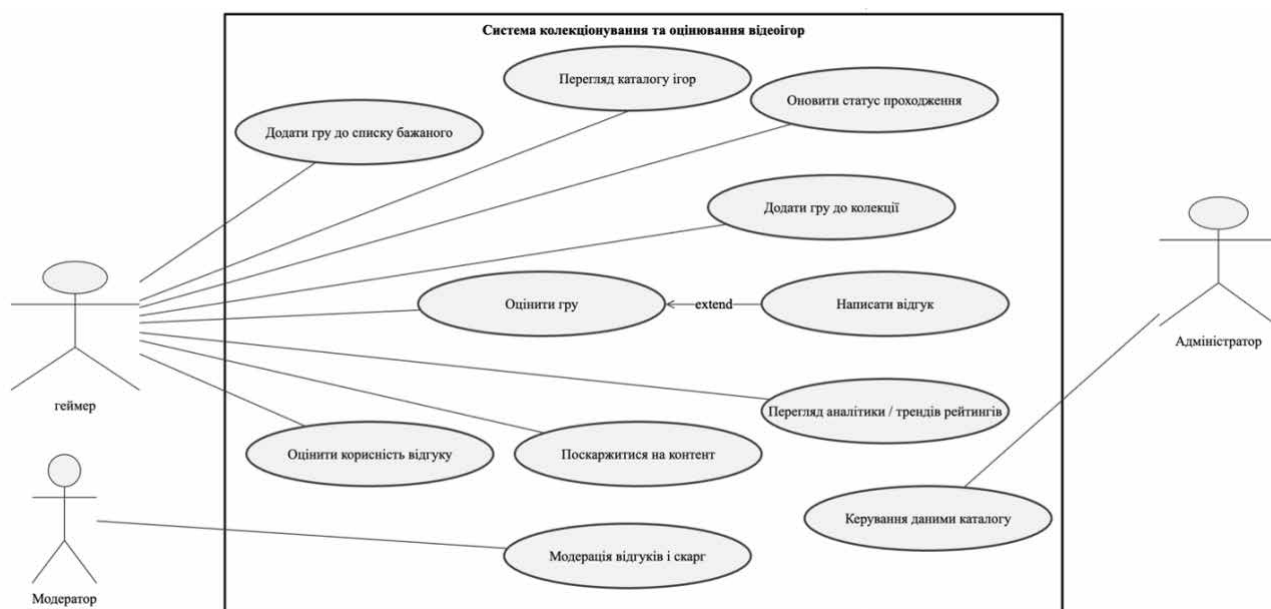


Рисунок 1.7 - Діаграма прецедентів системи колекціонування та оцінювання відеоігор

Основні ролі та функції користувачів системи наведено в таблиці 1.3.

Таблиця 1.3

Ролі користувачів та основні функції системи

Роль користувача	Основні функції	Результат виконання
Геймер	Перегляд каталогу ігор, додавання гри до колекції, додавання до списку бажаного, оновлення статусу проходження	Сформована персональна колекція відеоігор
Геймер	Оцінювання гри, написання відгуку, оцінювання корисності відгуку	Збережені користувацькі оцінки та відгуки

Геймер	Перегляд аналітики та трендів рейтингів	Отримання рейтингової та статистичної інформації
Геймер	Подання скарги на некоректний контент	Передача скарги на модерацію

Продовження таблиці 1.3

Модератор	Модерація відгуків і скарг	Приховування, видалення або залишення контенту без змін
Адміністратор	Керування даними каталогу	Додавання, редагування або видалення відомостей про ігри
Адміністратор	Перегляд узагальнених показників	Контроль стану каталогу, рейтингів і активності користувачів

Діаграма прецедентів показує, що найбільша кількість функцій належить геймеру. Це пояснюється тим, що основна мета системи полягає у веденні персональної ігрової колекції та накопиченні користувацьких оцінок. Окремо виділено функцію написання відгуку, яка розширює процес оцінювання гри. Це дає змогу користувачеві не лише виставити числову оцінку, а й залишити текстове пояснення власного враження від гри.

Для уточнення логіки взаємодії між користувачами та програмними компонентами побудовано діаграму послідовності, наведену на рисунку 1.8. Вона відображає основний сценарій роботи геймера з каталогом і колекцією, а також окремі сценарії модерації та адміністрування. У моделі виділено об'єкти: геймер, система, каталог ігор, колекція, рейтинги і відгуки, модератор, адміністратор та адмінпанель.

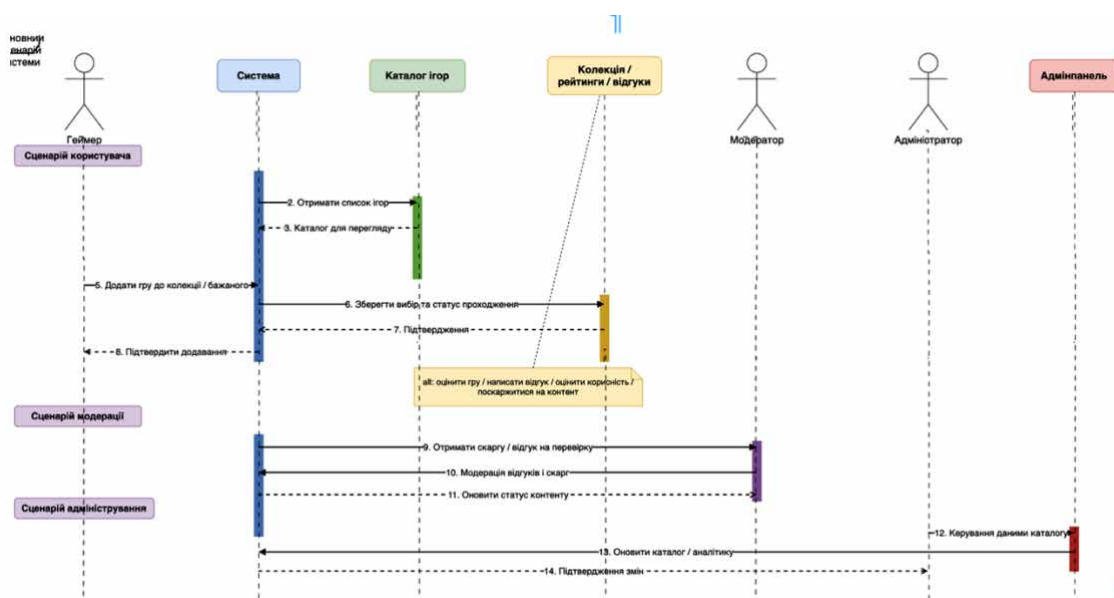


Рисунок 1.8 - Діаграма послідовності процесу роботи із системою

Основний сценарій починається з дії геймера, який звертається до системи для отримання списку ігор. Система надсилає запит до каталогу ігор, після чого повертає користувачеві перелік доступних записів. Після вибору гри користувач може додати її до колекції або списку бажаного. Система передає відповідні дані до модуля колекцій, де зберігається вибір користувача, статус проходження та інші параметри. Після успішного збереження користувач отримує підтвердження про виконання операції.

Окремий фрагмент діаграми описує дії, пов'язані з оцінюванням гри, написанням відгуку, оцінюванням корисності відгуку або поданням скарги на контент. Такі операції передаються до модуля колекцій, рейтингів і відгуків, де зберігаються оцінки, текстові коментарі та службові позначки. У разі надходження скарги модератор отримує відгук або скаргу на перевірку, виконує модерацію та оновлює статус контенту.

Адміністративний сценарій передбачає звернення адміністратора до адмінпанелі для керування даними каталогу. Адміністратор може змінювати відомості про відеоігри, жанри, платформи, розробників і видавців. Після виконання операції система оновлює каталог або аналітичні дані та повертає підтвердження про внесення змін.

Логіка виконання основних дій у системі подана на рисунку 1.9 у вигляді діаграми активності. Діаграма має три доріжки відповідальності: геймер, модератор та адміністратор. Це дозволяє чітко розмежувати дії кожного учасника та показати, які операції виконуються залежно від ролі користувача.

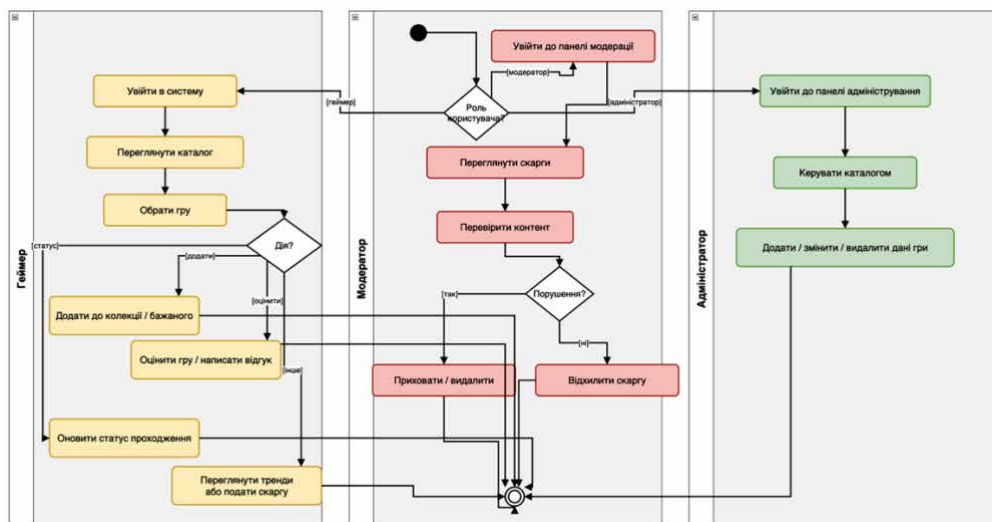


Рисунок 1.9 - Діаграма активності процесу роботи користувачів у системі

Процес починається з входу користувача в систему. Після автентифікації визначається роль користувача. Якщо користувач має роль геймера, він переходить до перегляду каталогу, вибирає гру та виконує одну з доступних дій: додає гру до колекції або списку бажаного, оцінює гру, пише відгук, оновлює статус проходження, переглядає тренди або подає скаргу. Якщо користувач має роль модератора, він переходить до панелі модерації, переглядає скарги, перевіряє контент і приймає рішення щодо наявності порушення. У разі підтвердження порушення контент приховується або видаляється, інакше скарга відхиляється.

Для адміністратора передбачено окремий сценарій роботи. Після входу до панелі адміністрування він переходить до керування каталогом і може додавати, змінювати або видаляти дані гри. Такі дії впливають на наповнення каталогу та забезпечують актуальність довідкової інформації в системі.

Узагальнена характеристика основних процесів системи наведена в таблиці 1.4.

Таблиця 1.4

Основні процеси предметної області

Процес	Учасник	Вхідні дані	Вихідні дані
Перегляд каталогу ігор	Геймер	Запит на перегляд каталогу	Список ігор з основними характеристиками

Додавання гри до колекції	Геймер	Обрана гра, статус, тип копії, примітки	Новий елемент колекції користувача
Оновлення статусу проходження	Геймер	Ідентифікатор гри, новий статус, години гри	Оновлений запис у колекції
Оцінювання гри	Геймер	Оцінка від 1 до 10, ідентифікатор гри	Збережена оцінка користувача
Написання відгуку	Геймер	Текст відгуку, оцінка, дані гри	Опублікований або очікуваний на перевірку відгук
Подання скарги	Геймер	Ідентифікатор відгуку, причина скарги	Скарга для модератора
Модерація контенту	Модератор	Скарги, відгуки, службові позначки	Оновлений статус контенту
Керування каталогом	Адміністратор	Дані гри, жанру, платформи, розробника, видавця	Оновлений каталог відеоігор

Побудовані моделі демонструють, що розроблювана система має чітко виражену модульну структуру. Користувацький функціонал зосереджений навколо каталогу, колекції та рейтингового механізму. Модераційний функціонал забезпечує контроль якості контенту, а адміністративний функціонал відповідає за актуальність довідкових даних. Такий розподіл ролей дає змогу зменшити дублювання операцій, підвищити контрольованість даних і забезпечити логічну послідовність роботи системи.

Отже, моделювання предметної області дозволило визначити основних учасників системи, сформувати перелік функціональних можливостей, описати сценарії взаємодії та уточнити логіку виконання основних процесів. Отримані UML-моделі є основою для подальшого проєктування архітектури програмного забезпечення, структури бази даних і програмних модулів системи колекціонування та оцінювання відеоігор.

1.4 Аналіз вимог інформаційної системи

Аналіз вимог програмної системи виконано з метою визначення функцій, обмежень і характеристик, які повинна забезпечувати система колекціонування та оцінювання відеоігор. Вимоги сформовано з урахуванням ролей користувачів,

логіки предметної області, побудованих UML-моделей та структури майбутньої бази даних.

Розроблювана система має забезпечувати роботу трьох основних ролей: геймера, модератора та адміністратора. Геймер використовує систему для перегляду каталогу, формування власної колекції, оновлення статусу проходження, оцінювання ігор і написання відгуків. Модератор відповідає за контроль користувацького контенту, перевірку скарг і прийняття рішень щодо відгуків. Адміністратор керує каталогом ігор, довідниками жанрів, платформ, розробників і видавців.

Основні вимоги користувачів до системи наведено в таблиці 1.5.

Таблиця 1.5

Вимоги користувачів до програмної системи

Роль	Потреба користувача	Очікуваний результат
Геймер	Переглядати каталог відеоігор	Отримання списку ігор з основними характеристиками
Геймер	Додавати гру до особистої колекції	Створення запису про гру в колекції користувача
Геймер	Додавати гру до списку бажаного	Збереження гри для подальшого проходження або придбання
Геймер	Змінювати статус проходження	Актуальний стан гри в особистій колекції
Геймер	Оцінювати гру	Збереження оцінки та участь у формуванні рейтингу
Геймер	Писати відгук	Додавання текстового опису власного досвіду
Геймер	Переглядати рейтинг і статистику	Отримання аналітичної інформації про популярні ігри
Геймер	Подавати скаргу на контент	Передача проблемного відгуку на модерацію
Модератор	Перевіряти скарги та відгуки	Контроль якості користувацького контенту
Адміністратор	Керувати каталогом ігор	Актуалізація даних про ігри, жанри, платформи та видавців

Функціональні вимоги визначають набір операцій, які повинна виконувати система. Вони охоплюють автентифікацію, роботу з каталогом, керування колекціями, оцінювання, модерацію та адміністрування даних.

Таблиця 1.6

Функціональні вимоги до програмної системи

Код	Вимога	Опис реалізації
F1	Реєстрація та вхід користувача	Система повинна забезпечувати створення облікового запису, вхід у систему та перевірку даних користувача
F2	Керування профілем	Користувач повинен мати можливість переглядати та оновлювати власні дані
F3	Перегляд каталогу відеоігор	Система повинна відображати список ігор з назвою, роком випуску, описом, жанром, платформою, розробником і видавцем
F4	Пошук і фільтрація ігор	Користувач повинен мати можливість шукати ігри за назвою, жанром, платформою, роком випуску або рейтингом
F5	Додавання гри до колекції	Система повинна дозволяти додавати гру до особистої колекції користувача
F6	Ведення списку бажаного	Користувач повинен мати можливість додавати гру до списку бажаного

Продовження таблиці 1.6

F7	Оновлення статусу проходження	Система повинна підтримувати статуси гри, наприклад заплановано, грається, пройдено, призупинено, покинуто
F8	Облік цифрових і фізичних копій	Система повинна зберігати тип копії гри та додаткові атрибути для цифрового або фізичного варіанта
F9	Оцінювання гри	Користувач повинен мати можливість виставити грі оцінку від 1 до 10
F10	Написання відгуку	Система повинна дозволяти створювати текстовий відгук до гри
F11	Оцінювання корисності відгуку	Користувач повинен мати можливість позначити відгук як корисний або некорисний
F12	Подання скарги	Система повинна дозволяти подати скаргу на некоректний відгук або інший користувацький контент
F13	Модерація контенту	Модератор повинен мати можливість переглядати скарги, приховувати або видаляти некоректні відгуки
F14	Керування каталогом	Адміністратор повинен мати можливість додавати, редагувати та видаляти дані про ігри
F15	Формування рейтингу	Система повинна обчислювати середню оцінку гри на основі користувацьких рейтингів

F16	Перегляд аналітики	Система повинна відображати рейтинги, популярні жанри, кількість оцінок, кількість відгуків і статистику колекцій
-----	--------------------	---

Нефункціональні вимоги визначають якісні характеристики програмної системи. Вони стосуються продуктивності, зручності, надійності, масштабованості та супроводу програмного забезпечення.

Таблиця 1.7

Нефункціональні вимоги до програмної системи

Код	Вимога	Характеристика
NF1	Продуктивність	Основні сторінки системи повинні відкриватися без помітних затримок, час відповіді для типових операцій має становити 1-2 секунди
NF2	Зручність інтерфейсу	Інтерфейс має бути простим, зрозумілим і не перевантаженим зайвими елементами
NF3	Доступність	Система повинна працювати через сучасний веббраузер без встановлення додаткового клієнтського ПЗ
NF4	Надійність	Система повинна коректно обробляти помилки введення, дублювання записів і некоректні запити
NF5	Масштабованість	Архітектура системи повинна дозволяти додавання нових модулів, наприклад рекомендацій або імпорту з інших платформ
NF6	Супроводжуваність	Програмний код має бути розділений на логічні модулі: автентифікація, каталог, колекції, рейтинги, модерація

Продовження таблиці 1.7

NF7	Цілісність даних	База даних повинна забезпечувати зв'язки між користувачами, іграми, колекціями, оцінками та відгуками
NF8	Переносимість	Серверна частина повинна мати можливість розгортання на локальному сервері або хмарній інфраструктурі

Вимоги до інформаційної безпеки є важливими, оскільки система працює з обліковими записами користувачів, паролями, оцінками, відгуками та службовими даними. Основні безпекові вимоги наведено в таблиці 1.8.

Таблиця 1.8

Вимоги до інформаційної безпеки

Код	Вимога	Опис
S1	Захист облікових записів	Паролі користувачів повинні зберігатися у вигляді хешу

S2	Автентифікація користувачів	Доступ до персональної колекції та службових функцій має надаватися лише після входу в систему
S3	Розмежування прав доступу	Геймер, модератор і адміністратор повинні мати різні права доступу
S4	Захист адміністративних функцій	Керування каталогом має бути доступним лише адміністратору
S5	Захист модераційних функцій	Перевірка скарг і зміна статусу відгуків має бути доступна лише модератору або адміністратору
S6	Валідація вхідних даних	Система повинна перевіряти оцінки, текстові поля, статуси, email та інші дані перед збереженням
S7	Захист від некоректних дій	Користувач не повинен мати можливості змінювати чужі колекції, оцінки та відгуки
S8	Журналювання службових операцій	Бажано фіксувати дії адміністратора та модератора для контролю змін

Технічні вимоги визначають програмні засоби та архітектурні рішення, які доцільно використати під час реалізації системи. З урахуванням попереднього проєктування система може будуватися за клієнт-серверною архітектурою з використанням вебклієнта, серверного API та реляційної бази даних.

Таблиця 1.9

Технічні вимоги до програмної системи

Код	Вимога	Реалізація
T1	Тип системи	Веборієнтована клієнт-серверна система
T2	Клієнтська частина	HTML, CSS, JavaScript або інший сучасний вебінтерфейс
T3	Серверна частина	Python API на основі FastAPI

Продовження таблиці 1.9

T4	Формат обміну даними	HTTPS/JSON, REST-підхід
T5	База даних	PostgreSQL
T6	Робота з даними	SQLAlchemy для доступу до БД, Alembic для міграцій
T7	Перевірка даних	Pydantic для валідації вхідних і вихідних структур
T8	Авторизація	JWT або сесійний механізм доступу
T9	Середовище розгортання	Локальний сервер або хмарний VPS
T10	Резервне копіювання	Періодичне створення копій бази даних

Вхідними даними системи є облікові дані користувачів, інформація про відеоігри, жанри, платформи, розробників, видавців, записи колекцій, статуси

проходження, оцінки, тексти відгуків, скарги та службові дії адміністратора. Вихідними даними є персональна колекція користувача, список бажаного, оновлений каталог ігор, середні рейтинги, відгуки, результати модерації та аналітичні показники.

Узагальнений перелік основних даних системи наведено в таблиці 1.10.

Таблиця 1.10

Вхідні та вихідні дані програмної системи

Група даних	Приклади даних	Призначення
Облікові дані	Логін, email, пароль, роль	Вхід до системи та розмежування доступу
Дані каталогу	Назва гри, рік випуску, опис, розробник, видавець	Формування каталогу відеоігор
Довідкові дані	Жанри, платформи, типи копій, статуси	Класифікація ігор і колекцій
Дані колекції	Обрана гра, статус, години, примітки	Ведення особистої бібліотеки користувача
Дані оцінювання	Оцінка, текст відгуку, дата оцінювання	Формування рейтингу та користувацьких відгуків
Дані модерації	Скарга, причина, статус перевірки	Контроль якості користувацького контенту
Аналітичні дані	Середній рейтинг, кількість оцінок, популярні жанри	Перегляд статистики та трендів

Отже, програмна система повинна забезпечувати повний цикл роботи з відеоіграми: від перегляду каталогу та додавання гри до колекції до оцінювання, написання відгуків, модерації контенту та формування рейтингової інформації. Сформовані вимоги є основою для подальшого проектування бази даних, архітектури програмного забезпечення, інтерфейсу користувача та тестування системи.

1.5 Постановка завдання

На основі аналізу предметної області, розгляду існуючих програмних рішень, сформованих вимог та результатів UML-моделювання було визначено

постановку завдання для розроблення програмного забезпечення з колекціонування та оцінювання відеоігор за рейтингом користувачів.

Завдання полягає у проєктуванні та розробленні інформаційної системи, призначеної для централізованого ведення каталогу відеоігор, формування персональних ігрових колекцій, збереження статусів проходження, оцінок, відгуків та формування рейтингової інформації на основі користувацьких даних.

Проєктована система розглядається як веб-орієнтований програмний комплекс із клієнт-серверною архітектурою. Клієнтська частина забезпечує взаємодію користувача з каталогом, колекцією, рейтингами та відгуками, а серверна частина відповідає за оброблення запитів, перевірку даних, керування бізнес-логікою та взаємодію з базою даних. Для зберігання інформації передбачається використання реляційної бази даних, у якій фіксуються відомості про користувачів, відеоігри, жанри, платформи, розробників, видавців, елементи колекцій, оцінки та відгуки.

Система повинна забезпечувати повний цикл роботи з відеогрою в межах користувацької колекції: від перегляду каталогу та вибору гри до додавання її в особисту бібліотеку, визначення статусу проходження, зазначення типу копії, виставлення оцінки, написання відгуку та участі у формуванні загального рейтингу.

Для досягнення поставленої мети необхідно спроектувати програмну систему, що складається з таких функціональних підсистем:

1. Підсистема користувацької взаємодії, яка забезпечує роботу геймера, модератора та адміністратора через веб-інтерфейс.
2. Підсистема автентифікації та керування доступом, що відповідає за реєстрацію, вхід користувачів, перевірку ролей та обмеження доступу до службових функцій.
3. Підсистема ведення каталогу відеоігор, яка забезпечує зберігання та редагування даних про ігри, жанри, платформи, розробників і видавців.

4. Підсистема керування колекціями, що дозволяє користувачам додавати ігри до особистої колекції, змінювати статус проходження, вказувати кількість годин, тип копії та додаткові примітки.

5. Підсистема оцінювання та відгуків, яка забезпечує виставлення оцінок, написання текстових відгуків, оцінювання корисності відгуків та збереження історії користувацьких оцінок.

6. Підсистема модерації, що відповідає за перевірку скарг, контроль якості користувацького контенту, приховування або видалення некоректних відгуків.

7. Підсистема аналітики та формування рейтингу, яка обчислює середні оцінки ігор, кількість відгуків, популярність жанрів, активність користувачів і формує статистичні показники.

Основною метою постановки завдання є створення програмного забезпечення, яке забезпечує зручне, структуроване та контрольоване ведення ігрових колекцій користувачів. Розроблена система повинна усунути фрагментарність зберігання інформації про відеоігри, забезпечити централізований облік цифрових і фізичних копій, підтримати механізм оцінювання та надати користувачам можливість переглядати рейтингову й аналітичну інформацію.

Результатом виконання поставленого завдання має стати програмний продукт, який поєднує каталог відеоігор, персональні колекції, систему оцінювання, відгуки, модерацію та аналітику в єдиному інформаційному середовищі. Така система може бути використана геймерами, колекціонерами та адміністраторами ігрових каталогів для впорядкування даних про відеоігри, аналізу користувацьких оцінок і формування об'єктивного рейтингу ігор.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER-діаграми

Логічну модель даних розроблено для визначення основних інформаційних об'єктів системи колекціонування та оцінювання відеоігор, їх атрибутів і зв'язків між ними. Модель відображає структуру зберігання даних про користувачів, відеоігри, колекції, оцінки, жанри, платформи, розробників і видавців.

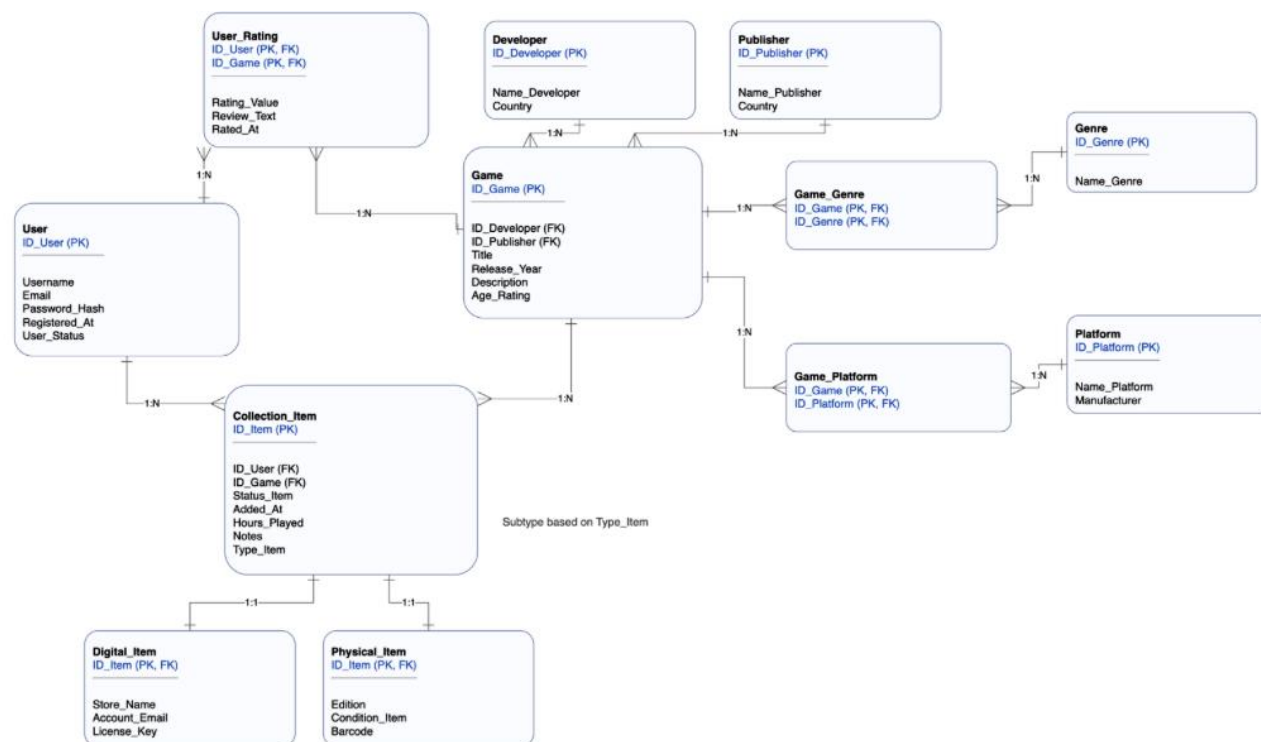


Рисунок 2.1 - Логічна модель даних системи колекціонування та оцінювання відеоігор

Центральною сутністю моделі є Game, яка містить основні відомості про відеогру: ідентифікатор, назву, рік випуску, опис і віковий рейтинг. Кожна гра пов'язана з одним розробником Developer та одним видавцем Publisher. Такий поділ дозволяє уникнути дублювання назв компаній у записах ігор і забезпечує цілісність довідкових даних.

Для класифікації відеоігор у моделі передбачено сутності Genre та Platform. Оскільки одна гра може належати до кількох жанрів і виходити на кількох платформах, зв'язки багато-до-багатьох реалізовано через проміжні сутності Game_Genre та Game_Platform. Це забезпечує гнучке розширення каталогу без порушення структури основної таблиці Game.

Користувачі системи представлені сутністю User, яка зберігає дані облікового запису: ім'я користувача, email, хеш пароля, дату реєстрації та статус. Зв'язок користувача з відеоіграми реалізовано через сутність Collection_Item. Вона фіксує, яку гру додано до колекції, який статус проходження встановлено, скільки годин зіграно, коли гру було додано та які примітки залишив користувач.

Оцінювання відеоігор реалізовано через сутність User_Rating. Вона пов'язує користувача з конкретною грою та містить числову оцінку, текст відгуку і дату оцінювання. Такий підхід дозволяє зберігати персональні оцінки користувачів і на їх основі формувати середній рейтинг гри.

Для підтримки різних типів копій ігор у моделі використано підтипи Digital_Item та Physical_Item. Вони наслідують ідентифікатор елемента колекції та зберігають специфічні дані. Для цифрової копії це назва магазину, email акаунта та ліцензійний ключ. Для фізичної копії це видання, стан носія та штрихкод.

Таблиця 2.1

Основні сутності логічної моделі даних

Сутність	Призначення	Основні атрибути
User	Зберігання даних користувачів системи	ID_User, Username, Email, Password_Hash, Registered_At, User_Status
Game	Зберігання основної інформації про відеогру	ID_Game, Title, Release_Year, Description, Age_Rating
Developer	Зберігання даних про розробників ігор	ID_Developer, Name_Developer, Country
Publisher	Зберігання даних про видавців ігор	ID_Publisher, Name_Publisher, Country
Genre	Довідник жанрів відеоігор	ID_Genre, Name_Genre

Platform	Довідник ігрових платформ	ID_Platform, Name_Platform, Manufacturer
----------	---------------------------	--

Продовження таблиці 2.1

Game_Genre	Зв'язок між іграми та жанрами	ID_Game, ID_Genre
Game_Platform	Зв'язок між іграми та платформами	ID_Game, ID_Platform
Collection_Item	Елемент персональної колекції користувача	ID_Item, ID_User, ID_Game, Status_Item, Added_At, Hours_Played, Notes, Type_Item
Digital_Item	Дані про цифрову копію гри	ID_Item, Store_Name, Account_Email, License_Key
Physical_Item	Дані про фізичну копію гри	ID_Item, Edition, Condition_Item, Barcode
User_Rating	Зберігання оцінок і відгуків користувачів	ID_User, ID_Game, Rating_Value, Review_Text, Rated_At

Логічна модель даних приведена до третьої нормальної форми. На першому етапі нормалізації забезпечено атомарність атрибутів: кожне поле містить лише одне значення, наприклад окремо зберігаються назва гри, рік випуску, жанр, платформа, рейтинг і текст відгуку. Повторювані групи даних винесено в окремі таблиці.

На другому етапі нормалізації усунуто часткові залежності. У проміжних таблицях Game_Genre, Game_Platform та User_Rating використано складені ключі, а всі неключові атрибути залежать від повного ключа. Наприклад, оцінка користувача залежить одночасно від користувача і гри, а не лише від одного з цих ідентифікаторів.

На третьому етапі нормалізації усунуто транзитивні залежності. Дані про розробників, видавців, жанри та платформи не зберігаються безпосередньо в таблиці Game, а винесені в окремі сутності. Це зменшує дублювання даних, спрощує оновлення довідкової інформації та підвищує цілісність бази даних.

Таблиця 2.2

Перевірка нормалізації логічної моделі

Нормальна форма	Реалізація в моделі	Результат
1НФ	Усі атрибути мають атомарні значення	Відсутні повторювані групи даних
2НФ	Неключові атрибути залежать від повного первинного ключа	Коректно реалізовано зв'язки користувач-гра, гра-жанр, гра-платформа

Продовження таблиці 2.2

3НФ	Довідкові дані винесено в окремі сутності	Усунуто дублювання розробників, видавців, жанрів і платформ
-----	---	---

Запропонована ER-модель забезпечує цілісне зберігання даних і підтримує основні функції майбутньої системи: ведення каталогу відеоігор, формування персональних колекцій, розмежування цифрових і фізичних копій, збереження оцінок і відгуків, а також формування рейтингу на основі користувацьких даних. Така структура є придатною для подальшого переходу до фізичної моделі бази даних і реалізації SQL-структури.

2.2 Діаграма класів і кооперації

Діаграму класів побудовано для визначення основних об'єктів програмної системи, їх атрибутів, операцій та зв'язків. Модель відображає логіку роботи системи на рівні програмних класів і є основою для подальшої реалізації модулів каталогу ігор, користувацьких колекцій, оцінювання та відгуків.

На рисунку 2.2 наведено діаграму класів системи колекціонування та оцінювання відеоігор.

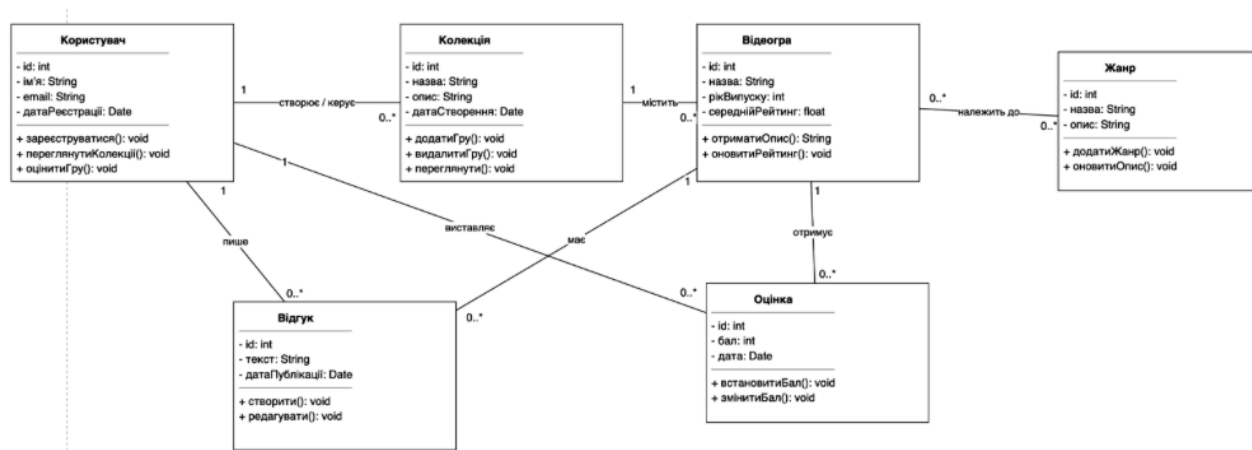


Рисунок 2.2 - Діаграма класів системи колекціонування та оцінювання відеоігор

Основними класами системи є Користувач, Колекція, Відеогра, Жанр, Оцінка та Відгук. Клас Користувач зберігає ідентифікатор, ім'я, email та дату реєстрації, а також містить операції реєстрації, перегляду колекції та оцінювання гри. Клас Колекція відповідає за збереження персонального набору ігор користувача, має назву, опис, дату створення та операції додавання, видалення й перегляду ігор.

Клас Відеогра є центральним об'єктом моделі. Він містить назву гри, рік випуску та середній рейтинг. Із ним пов'язані класи Жанр, Оцінка та Відгук. Клас Жанр використовується для класифікації ігор, клас Оцінка фіксує числовий бал користувача, а клас Відгук зберігає текст рецензії та дату публікації. Така структура дозволяє відокремити довідкову інформацію про ігри від користувацьких даних і забезпечити коректне формування рейтингу.

Таблиця 2.3

Основні класи програмної системи

Клас	Призначення	Основні атрибути	Основні операції
Користувач	Представляє зареєстрованого користувача системи	id, ім'я, email, датаРеєстрації	зареєструватися, переглянутиКолекцію, оцінитиГру
Колекція	Зберігає перелік ігор користувача	id, назва, опис, датаСтворення	датиГру, видалитиГру, переглянути

Відеогра	Містить основні дані про гру	id, назва, рікВипуску, середнійРейтинг	отриматиОпис, оновитиРейтинг
Жанр	Забезпечує класифікацію відеоігор	id, назва, опис	додатиЖанр, оновитиОпис
Оцінка	Фіксує числову оцінку гри	id, бал, дата	встановитиБал, змінитиБал
Відгук	Зберігає текстовий відгук користувача	id, текст, датаПублікації	створити, редагувати

Зв'язки між класами відповідають логіці предметної області. Один користувач може створювати або керувати власною колекцією. Колекція містить набір відеоігор, а одна відеогра може входити до різних колекцій. Користувач може залишати відгуки та виставляти оцінки, а кожна відеогра отримує набір оцінок, на основі яких обчислюється середній рейтинг. Клас Жанр пов'язаний із класом Відеогра та використовується для групування ігор за тематикою.

Для уточнення взаємодії між класами побудовано три кооперації, що відображають основні сценарії роботи системи. На рисунку 2.3 подано кооперацію формування колекції.

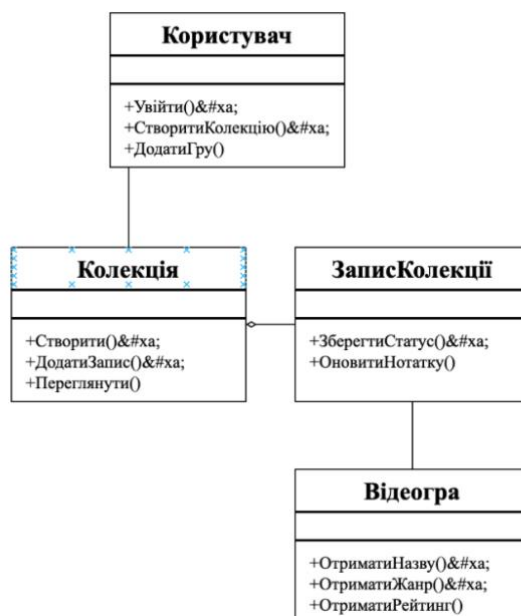


Рисунок 2.3 - Кооперація формування ігрової колекції користувача

У цій кооперації користувач входить у систему, створює колекцію та додає до неї відеогра. Запис колекції використовується як проміжний об'єкт, що

зберігає стан гри в межах конкретної колекції користувача. Через нього можна оновити статус проходження або додати нотатку.

На рисунку 2.4 наведено кооперацію класифікації відеоігор.

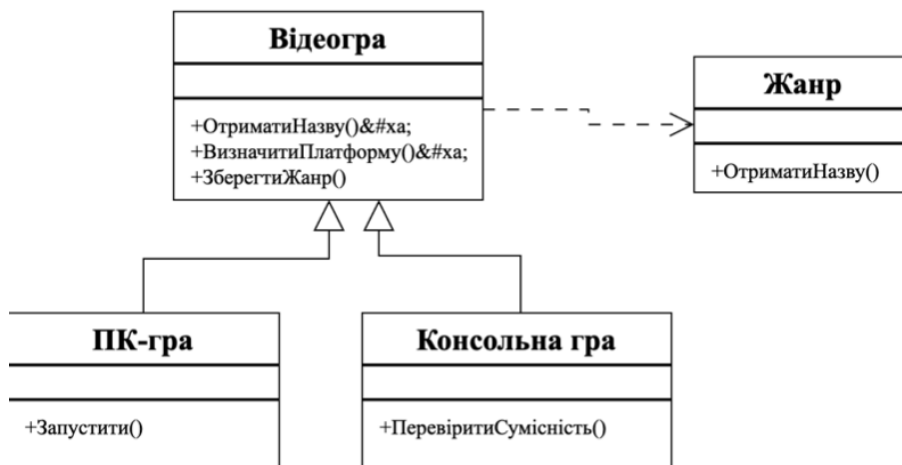


Рисунок 2.4 - Кооперація класифікації відеоігор

Кооперація класифікації показує, що відеогра пов'язується з жанром і може мати спеціалізовані підтипи, наприклад ПК-гру або консольну гру. Такий підхід дозволяє розширювати модель без зміни базового класу Відеогра та враховувати особливості різних типів ігор.

На рисунку 2.5 подано кооперацію оцінювання та роботи з відгуками.

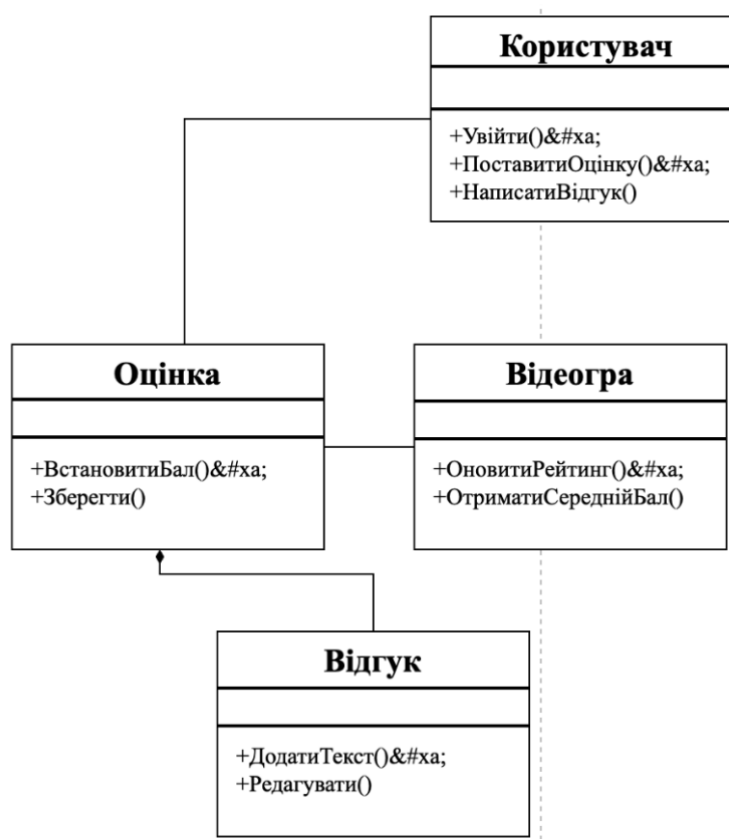


Рисунок 2.5 - Кооперація оцінювання відеоігор та додавання відгуків

У цій кооперації користувач виставляє оцінку відеогрі та може додати текстовий відгук. Оцінка зберігає числовий бал, а відгук містить текстову характеристику гри. Після збереження оцінки система оновлює рейтинг відеогри та забезпечує отримання середнього бала.

Побудована діаграма класів і кооперації узгоджуються з логічною моделлю даних та функціональними вимогами системи. Вони відображають основні програмні об'єкти, зв'язки між ними та ключові сценарії роботи: формування колекції, класифікацію ігор, оцінювання та написання відгуків. Отримані моделі можуть бути використані як основа для реалізації серверної логіки, структури класів і модулів програмного забезпечення.

2.3 Діаграма компонентів розроблювальної системи

Діаграму компонентів розроблено для відображення загальної програмної структури системи колекціонування та оцінювання відеоігор. Вона показує

основні частини програмного забезпечення, їх призначення та спосіб взаємодії між клієнтською частиною, серверною логікою і базою даних.

На рисунку 2.6 подано діаграму компонентів розроблюваної системи.

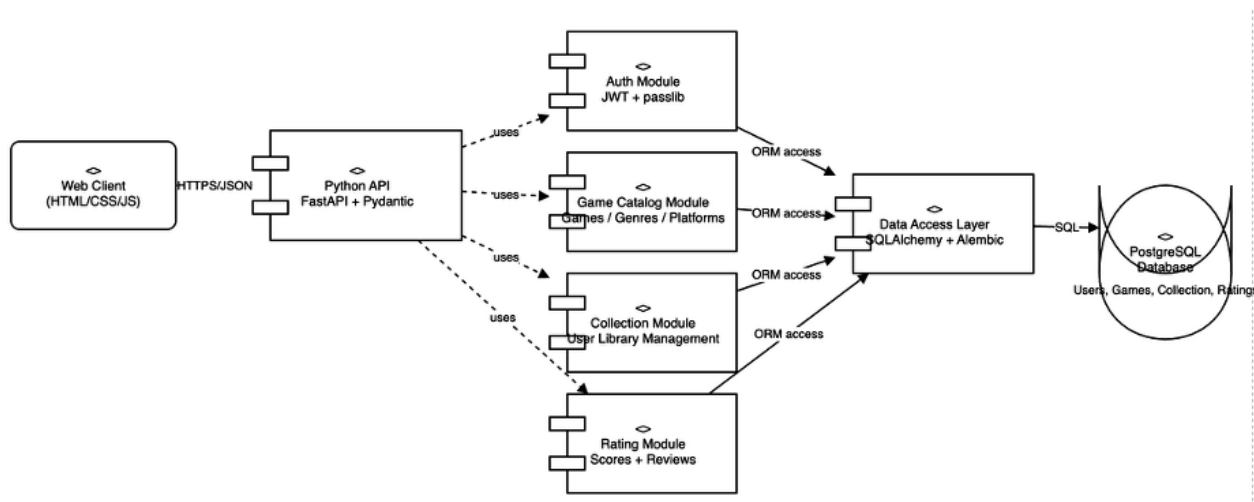


Рисунок 2.6 - Діаграма компонентів системи колекціонування та оцінювання відеоігор

Клієнтська частина представлена компонентом Web Client, який реалізується засобами HTML, CSS і JavaScript. Через цей компонент користувач взаємодіє із системою: переглядає каталог ігор, додає ігри до колекції, змінює статус проходження, виставляє оцінки та залишає відгуки. Обмін даними між клієнтом і сервером виконується через HTTPS/JSON.

Серверна частина представлена компонентом Python API, побудованим на основі FastAPI та Pydantic. Цей компонент приймає запити від клієнта, перевіряє вхідні дані, викликає відповідні функціональні модулі та повертає результат користувачу. Pydantic використовується для валідації структур даних, що зменшує кількість помилок під час оброблення запитів.

Функціональна логіка системи поділена на окремі модулі. Auth Module відповідає за автентифікацію користувачів, перевірку доступу та роботу з JWT-токенами. Для захисту паролів передбачено використання passlib. Game Catalog Module забезпечує роботу з каталогом ігор, жанрами та платформами. Collection Module реалізує функції керування персональною бібліотекою користувача.

Rating Module відповідає за збереження оцінок, відгуків і формування рейтингової інформації.

Окремим компонентом виділено Data Access Layer. Він забезпечує доступ функціональних модулів до бази даних і приховує деталі роботи з SQL-запитами. Для реалізації цього шару використовується SQLAlchemy, а для керування змінами структури бази даних передбачено Alembic. Такий підхід спрощує супровід системи та дозволяє змінювати структуру БД без порушення роботи основних модулів.

Дані системи зберігаються в PostgreSQL Database. У базі даних розміщуються відомості про користувачів, відеоігри, жанри, платформи, колекції, оцінки та відгуки. Звернення до бази даних здійснюється лише через шар доступу до даних, що підвищує контрольованість операцій і зменшує залежність бізнес-логіки від конкретної реалізації БД.

Побудована компонентна структура є модульною. Кожен компонент відповідає за окрему частину функціональності, тому систему можна доопрацьовувати без повної зміни архітектури. Наприклад, у майбутньому можна додати модуль рекомендацій, імпорт ігор з інших платформ або розширену аналітику, не змінюючи базову взаємодію між клієнтом, API та базою даних.

Отже, діаграма компонентів показує логічний поділ програмного забезпечення на клієнтський інтерфейс, серверний API, функціональні модулі, шар доступу до даних і базу даних. Така структура відповідає вимогам до масштабованості, підтримуваності та подальшого розвитку системи колекціонування та оцінювання відеоігор.

2.4 Діаграма пакетів

Діаграма пакетів відображає логічну організацію програмного забезпечення системи колекціонування та оцінювання відеоігор. Вона показує

поділ системи на клієнтську та серверну частини, а також залежності між основними пакетами.

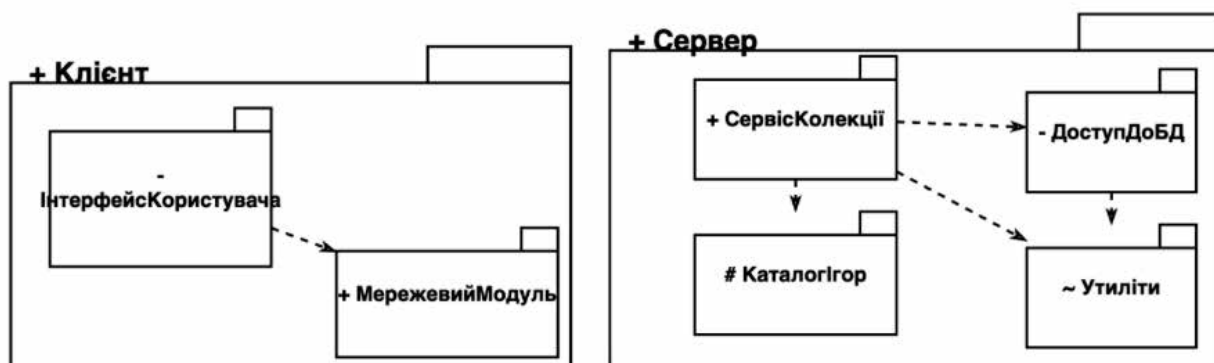


Рисунок 2.7 - Діаграма пакетів системи колекціонування та оцінювання відеоігор

У структурі системи виділено два головні пакети: Клієнт і Сервер. Пакет Клієнт відповідає за взаємодію користувача із системою. До нього входить ІнтерфейсКористувача, який забезпечує відображення каталогу, колекції, оцінок, відгуків і статистики, а також МережевийМодуль, що передає запити до серверної частини та приймає відповіді.

Пакет Сервер містить основну бізнес-логіку системи. СервісКолекції відповідає за роботу з ігровими колекціями, статусами проходження, оцінками та відгуками. КаталогІгор забезпечує оброблення даних про відеоігри, жанри, платформи, розробників і видавців. ДоступДоБД реалізує зв'язок із базою даних і виконує операції збереження, оновлення та вибірки інформації. Пакет Утиліти містить допоміжні функції для перевірки даних, оброблення помилок і роботи зі службовими параметрами.

Таблиця 2.4

Пакети програмної системи та їх призначення

Пакет	Розташування	Призначення
ІнтерфейсКористувача	Клієнт	Відображення сторінок, форм, каталогу, колекції, рейтингів і відгуків
МережевийМодуль	Клієнт	Передавання запитів до сервера та отримання відповідей
СервісКолекції	Сервер	Додавання ігор до колекції, зміна статусів, збереження оцінок і відгуків
КаталогІгор	Сервер	Робота з даними про ігри, жанри, платформи, розробників і видавців
ДоступДоБД	Сервер	Виконання запитів до бази даних та ізоляція роботи з БД від бізнес-логіки
Утиліти	Сервер	Валідація даних, оброблення помилок, службові функції

Залежності між пакетами побудовано так, щоб клієнтська частина не працювала з базою даних напряму. Усі запити користувача спочатку обробляються мережевим модулем, потім передаються до серверних сервісів, а доступ до даних здійснюється через окремий пакет ДоступДоБД. Це підвищує структурованість системи та спрощує її подальше розширення.

Отже, діаграма пакетів підтверджує модульну побудову системи. Такий поділ дозволяє окремо розробляти інтерфейс, серверну логіку, роботу з каталогом, колекціями та базою даних, що покращує супровід і тестування програмного забезпечення.

2.5 Висновки до другого розділу

У другому розділі було виконано проєктування інформаційного та програмного забезпечення системи колекціонування та оцінювання відеоігор за рейтингом користувачів. Основну увагу приділено побудові моделей, які визначають структуру даних, програмні класи, компоненти системи та логічний розподіл програмних пакетів.

У пункті 2.1 було розроблено логічну модель даних у вигляді ER-діаграми. У моделі визначено основні сутності системи: User, Game, Developer, Publisher, Genre, Platform, Collection_Item, Digital_Item, Physical_Item та User_Rating. Для реалізації зв'язків багато-до-багатьох між іграми, жанрами та платформами використано проміжні сутності Game_Genre і Game_Platform. Окремо виділено сутності для цифрових і фізичних копій ігор, що дозволяє гнучко описувати різні типи елементів користувацької колекції. Модель приведено до третьої нормальної форми, що забезпечує усунення дублювання даних, цілісність зв'язків і зручність подальшої реалізації бази даних.

У пункті 2.2 було побудовано діаграму класів і кооперації. Діаграма класів відображає основні програмні об'єкти системи: користувача, колекцію, відеогру, жанр, оцінку та відгук. Визначено атрибути, операції та зв'язки між класами. Побудовані кооперації уточнюють типові сценарії роботи системи: формування ігрової колекції, класифікацію відеоігор та оцінювання з додаванням відгуків. Це дозволило формалізувати взаємодію між об'єктами системи на рівні програмної логіки.

У пункті 2.3 було розроблено діаграму компонентів. Вона показує поділ системи на клієнтську частину, серверний API, функціональні модулі, шар доступу до даних і базу даних PostgreSQL. Клієнтський компонент забезпечує взаємодію користувача із системою через вебінтерфейс, а серверна частина реалізує автентифікацію, роботу з каталогом ігор, керування колекціями, оброблення оцінок і відгуків. Окреме виділення шару доступу до даних підвищує структурованість системи та спрощує супровід програмного забезпечення.

У пункті 2.4 було побудовано діаграму пакетів, яка відображає логічну організацію програмного коду. Систему поділено на клієнтський і серверний пакети. Клієнтська частина містить інтерфейс користувача та мережевий модуль, а серверна частина включає сервіс колекцій, каталог ігор, доступ до бази даних і допоміжні утиліти. Такий розподіл дозволяє окремо розробляти, тестувати та змінювати окремі частини системи без порушення роботи всього програмного комплексу.

Отже, у другому розділі було сформовано основні проєктні рішення для подальшої реалізації системи. Розроблені ER-модель, діаграма класів, кооперації, діаграма компонентів і діаграма пакетів узгоджуються між собою та відповідають функціональним вимогам системи. Отримані результати створюють основу для переходу до фізичного проєктування бази даних, реалізації серверної логіки, розроблення інтерфейсу користувача та тестування програмного забезпечення.

3 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ ТА РОЗРОБЛЕННЯ АЛГОРИТМІЧНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Обґрунтування вибору технологічних засобів та інструментів реалізації

Для реалізації програмної системи колекціонування та оцінювання відеоігор обрано веборієнтований підхід, оскільки він забезпечує доступ до системи через браузер, не потребує встановлення окремого клієнтського застосунку та добре відповідає клієнт-серверній архітектурі, визначеній у попередніх розділах.

Основою серверної частини обрано Node.js. Ця платформа дає змогу швидко створювати вебзастосунки, обробляти HTTP-запити, працювати з файлами, реалізовувати авторизацію користувачів і забезпечувати взаємодію між інтерфейсом та базою даних. Для маршрутизації, оброблення форм, сторінок каталогу, колекції, відгуків, модерації та адміністративної панелі використано Express.js.

Клієнтська частина реалізована засобами HTML, CSS і JavaScript. Такий набір технологій є достатнім для створення інтерактивного вебінтерфейсу з темною палітрою, картками ігор, формами реєстрації, фільтрами, пошуком, рейтингами та сторінками керування колекцією. CSS використано для оформлення інтерфейсу, адаптації елементів і створення єдиного візуального стилю системи.

Для зберігання даних у межах програмного прототипу використано локальну файлову базу даних у форматі JSON. У ній зберігаються користувачі, ігри, жанри, платформи, колекції, списки бажаного, оцінки, відгуки та скарги. Такий підхід спрощує запуск системи на будь-якому комп'ютері без додаткового встановлення СУБД. При цьому структура JSON-даних відповідає логіці раніше

спроектованої ER-моделі та може бути перенесена до PostgreSQL або іншої реляційної СУБД у разі подальшого розвитку системи.

Таблиця 3.1

Обґрунтування вибору технологічних засобів

Засіб	Призначення в системі	Обґрунтування вибору
Node.js	Серверна частина застосунку	Забезпечує швидку обробку запитів, роботу з файлами та запуск системи без складного налаштування
Express.js	Маршрутизація та серверна логіка	Дозволяє реалізувати сторінки каталогу, авторизацію, колекцію, відгуки, модерацію та адмінпанель
HTML	Структура сторінок	Використовується для побудови форм, таблиць, карток і сторінок системи
CSS	Візуальне оформлення	Забезпечує темний інтерфейс, адаптивність, зручне розміщення елементів і єдиний стиль
JavaScript	Клієнтська інтерактивність	Використовується для покращення взаємодії з формами, фільтрами та елементами інтерфейсу
JSON	Локальне зберігання даних	Дає змогу зберігати дані без окремої СУБД і швидко запускати прототип
npm	Керування залежностями	Забезпечує встановлення бібліотек і запуск проєкту однією командою
bcrypt або вбудований механізм хешування	Захист паролів	Дозволяє не зберігати паролі користувачів у відкритому вигляді
express-session	Сесії користувачів	Забезпечує підтримку авторизації та розмежування доступу за ролями

Архітектура реалізованої системи поділена на кілька логічних частин. Маршрути відповідають за оброблення запитів користувача, модулі роботи з даними виконують читання та оновлення JSON-бази, шаблони сторінок формують інтерфейс, а статичні файли відповідають за стилі, зображення та клієнтську логіку.

У системі передбачено три основні ролі: геймер, модератор і адміністратор. Геймер може реєструватися, входити в систему, переглядати

каталог ігор, додавати ігри до колекції, формувати список бажаного, виставляти оцінки й писати відгуки. Модератор переглядає скарги та керує проблемними відгуками. Адміністратор має доступ до керування каталогом відеоігор і може додавати, змінювати або видаляти записи.

Обраний стек технологій відповідає вимогам до навчального програмного продукту: система швидко запускається, не потребує складної інфраструктури, має зрозумілу структуру файлів, підтримує основні функції предметної області та може бути розширена в майбутньому. За потреби локальну JSON-базу можна замінити на PostgreSQL, а серверну частину доповнити REST API для роботи з мобільним застосунком або зовнішніми сервісами.

Отже, використання Node.js, Express.js, HTML, CSS, JavaScript і JSON-бази є доцільним для реалізації програмного прототипу системи колекціонування та оцінювання відеоігор. Такий вибір забезпечує простий запуск, достатню функціональність, зручність тестування та відповідність побудованим моделям системи.

3.2 Фізична модель даних розроблюваної системи

Фізичну модель даних розроблено на основі логічної ER-моделі системи колекціонування та оцінювання відеоігор. Вона визначає структуру таблиць, типи даних, первинні та зовнішні ключі, обмеження цілісності й індекси для прискорення пошуку.

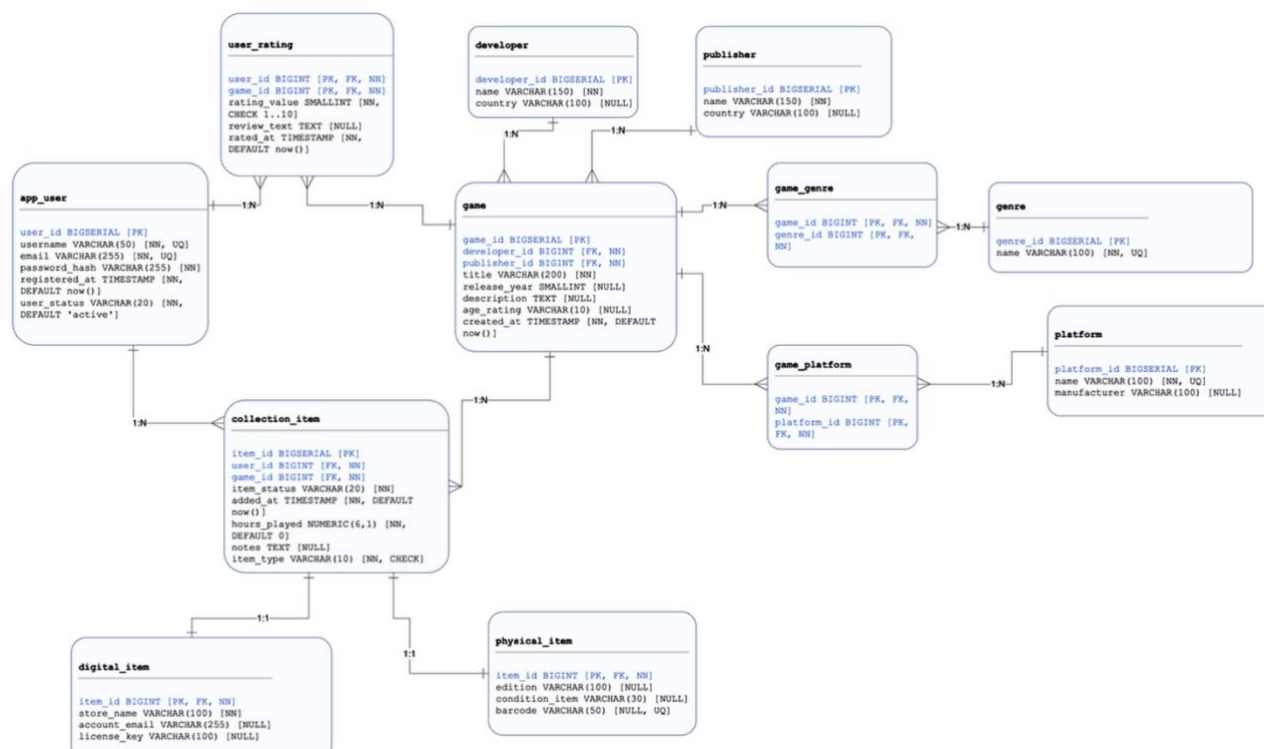


Рисунок 3.1 - Фізична модель даних системи колекціонування та оцінювання відеоігор

У фізичній моделі центральною таблицею є `game`, яка зберігає основні відомості про відеогру. Таблиці `developer`, `publisher`, `genre` та `platform` винесено окремо, що відповідає принципам нормалізації та усуває дублювання довідкових даних. Зв'язки багато-до-багатьох між іграми, жанрами та платформами реалізовано через проміжні таблиці `game_genre` і `game_platform`.

Для зберігання користувацьких даних використано таблицю `app_user`. Персональні колекції зберігаються в таблиці `collection_item`, яка пов'язує користувача з конкретною грою та містить статус проходження, дату додавання, кількість годин і тип копії. Окремі таблиці `digital_item` і `physical_item` деталізують цифрові та фізичні копії. Оцінки й відгуки користувачів зберігаються в таблиці `user_rating`.

Структура бази даних приведена до третьої нормальної форми. Усі атрибути є атомарними, довідкові дані винесено в окремі таблиці, а залежності між сутностями реалізовано через зовнішні ключі. Для контролю коректності даних використано CHECK-обмеження, UNIQUE-обмеження та індекси.

SQL-код створення фізичної моделі даних представлений у ДОДАТОК А

Для підвищення швидкодії пошуку та фільтрації даних створено індекси за полями, які найчастіше використовуються під час виконання запитів: назва гри, розробник, видавець, користувач, статус елемента колекції, дата оцінювання, жанр і платформа.

```
CREATE INDEX ix_game_title
  ON game(title);

CREATE INDEX ix_game_developer_id
  ON game(developer_id);

CREATE INDEX ix_game_publisher_id
  ON game(publisher_id);

CREATE INDEX ix_collection_item_user_id
  ON collection_item(user_id);

CREATE INDEX ix_collection_item_game_id
  ON collection_item(game_id);

CREATE INDEX ix_collection_item_status
  ON collection_item(item_status);

CREATE INDEX ix_user_rating_game_id
  ON user_rating(game_id);

CREATE INDEX ix_user_rating_rated_at
  ON user_rating(rated_at);

CREATE INDEX ix_game_genre_genre_id
  ON game_genre(genre_id);

CREATE INDEX ix_game_platform_platform_id
  ON game_platform(platform_id);
```

Для початкового наповнення довідників можуть використовуватися умовно-постійні дані. Вони забезпечують базову роботу каталогу, фільтрів і класифікації відеоігор.

```
INSERT INTO genre (name) VALUES
('Action'),
('Adventure'),
('RPG'),
('Shooter'),
('Strategy'),
('Racing'),
('Simulation'),
('Sports'),
('Horror'),
('Platformer');

INSERT INTO platform (name, manufacturer) VALUES
('PC', 'Microsoft'),
('PlayStation 5', 'Sony'),
('Xbox Series X/S', 'Microsoft'),
('Nintendo Switch', 'Nintendo'),
('PlayStation 4', 'Sony');
```

```
INSERT INTO developer (name, country) VALUES
('CD Projekt Red', 'Poland'),
('Rockstar Games', 'USA'),
('Valve', 'USA'),
('FromSoftware', 'Japan'),
('Nintendo', 'Japan');
```

```
INSERT INTO publisher (name, country) VALUES
('CD Projekt', 'Poland'),
('Rockstar Games', 'USA'),
('Valve', 'USA'),
('Bandai Namco Entertainment', 'Japan'),
('Nintendo', 'Japan');
```

Для отримання рейтингу гри на основі користувацьких оцінок доцільно використовувати агрегований запит. Він обчислює середню оцінку та кількість голосів для кожної відеогри.

```
SELECT
    g.game_id,
    g.title,
    ROUND(AVG(ur.rating_value), 2) AS average_rating,
    COUNT(ur.user_id) AS rating_count
FROM game g
LEFT JOIN user_rating ur ON ur.game_id = g.game_id
GROUP BY g.game_id, g.title
ORDER BY average_rating DESC NULLS LAST;
```

Розроблена фізична модель забезпечує збереження всіх основних даних системи: користувачів, відеоігор, жанрів, платформ, колекцій, цифрових і фізичних копій, оцінок та відгуків. Використання первинних і зовнішніх ключів забезпечує цілісність зв'язків, CHECK-обмеження контролюють допустимі значення, а індекси підвищують ефективність пошуку й аналітичних запитів. Така структура є придатною для реалізації серверної частини системи та подальшого розширення функціоналу.

3.3 Алгоритмізація програмних модулів системи

Алгоритмізація програмних модулів виконана для опису послідовності роботи основних частин системи колекціонування та оцінювання відеоігор. У межах розроблюваного програмного забезпечення виділено алгоритми роботи користувача з каталогом, додавання гри до колекції, оцінювання, оброблення HTTP-запитів, перевірки прав доступу та збереження даних.

На рисунку 3.2 наведено алгоритм додавання гри до колекції та подальшого оцінювання.

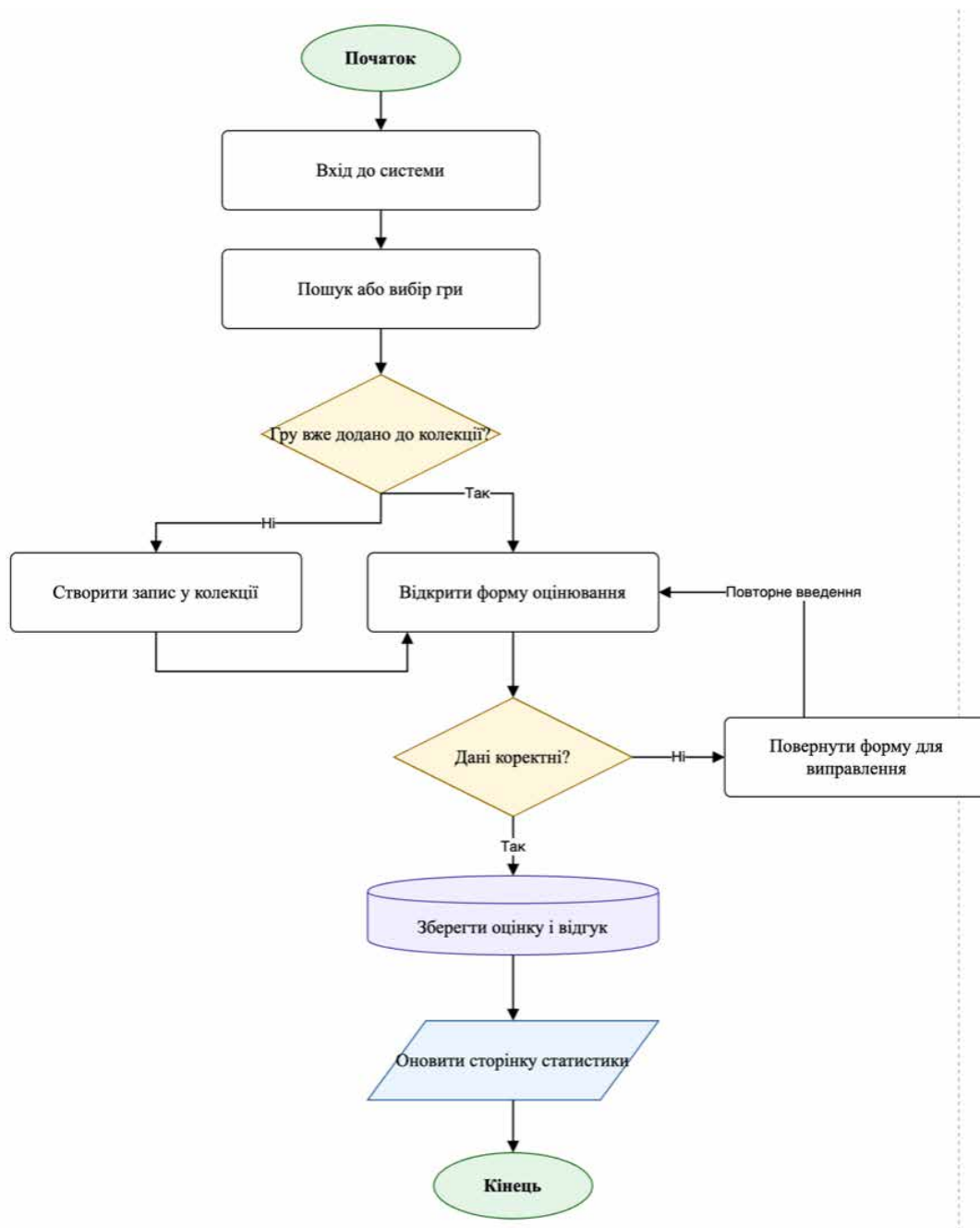


Рисунок 3.2 - Алгоритм додавання гри до колекції та оцінювання

Алгоритм починається з входу користувача до системи. Після авторизації користувач виконує пошук або вибір гри з каталогу. Далі система перевіряє, чи додана ця гра до особистої колекції. Якщо запису ще немає, створюється новий елемент колекції із зазначенням статусу проходження та додаткових параметрів. Якщо гра вже міститься в колекції, користувач переходить до форми оцінювання. Після введення бала та тексту відгуку система перевіряє коректність даних. У

разі помилки форма повертається для виправлення, а за правильного введення оцінка і відгук зберігаються, після чого оновлюється статистика гри.

На рисунку 3.3 подано алгоритм роботи користувацького сценарію в системі.

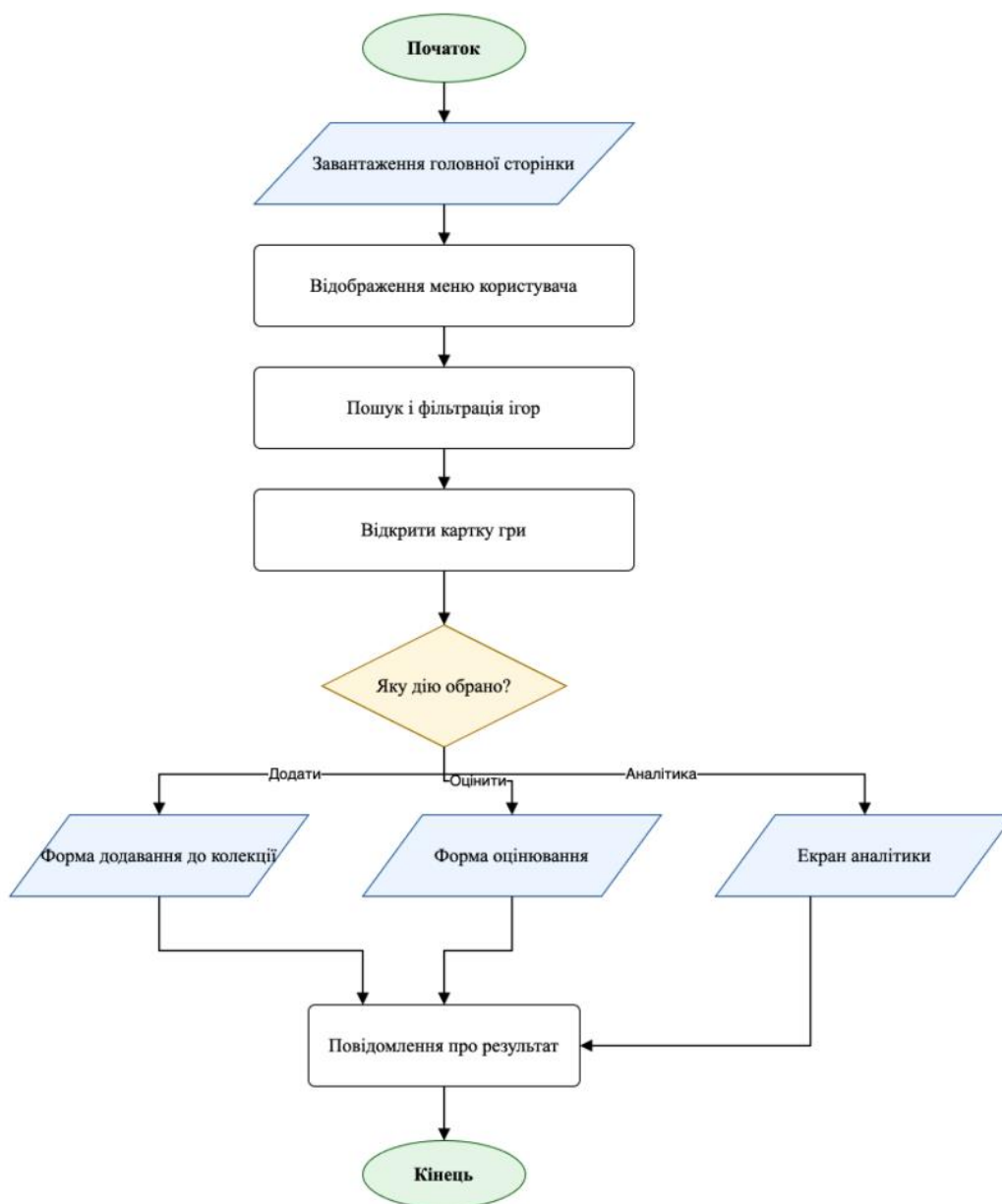


Рисунок 3.3 - Алгоритм користувацького сценарію роботи із системою

Після завантаження головної сторінки система відображає меню користувача. Далі користувач може виконати пошук і фільтрацію ігор, відкрити картку вибраної гри та обрати одну з доступних дій: додати гру до колекції, оцінити її або перейти до аналітики. Після виконання дії система формує

повідомлення про результат. Такий алгоритм забезпечує просту й послідовну взаємодію користувача з основними функціями програмного забезпечення.

На рисунку 3.4 зображено алгоритм роботи клієнтської частини системи.

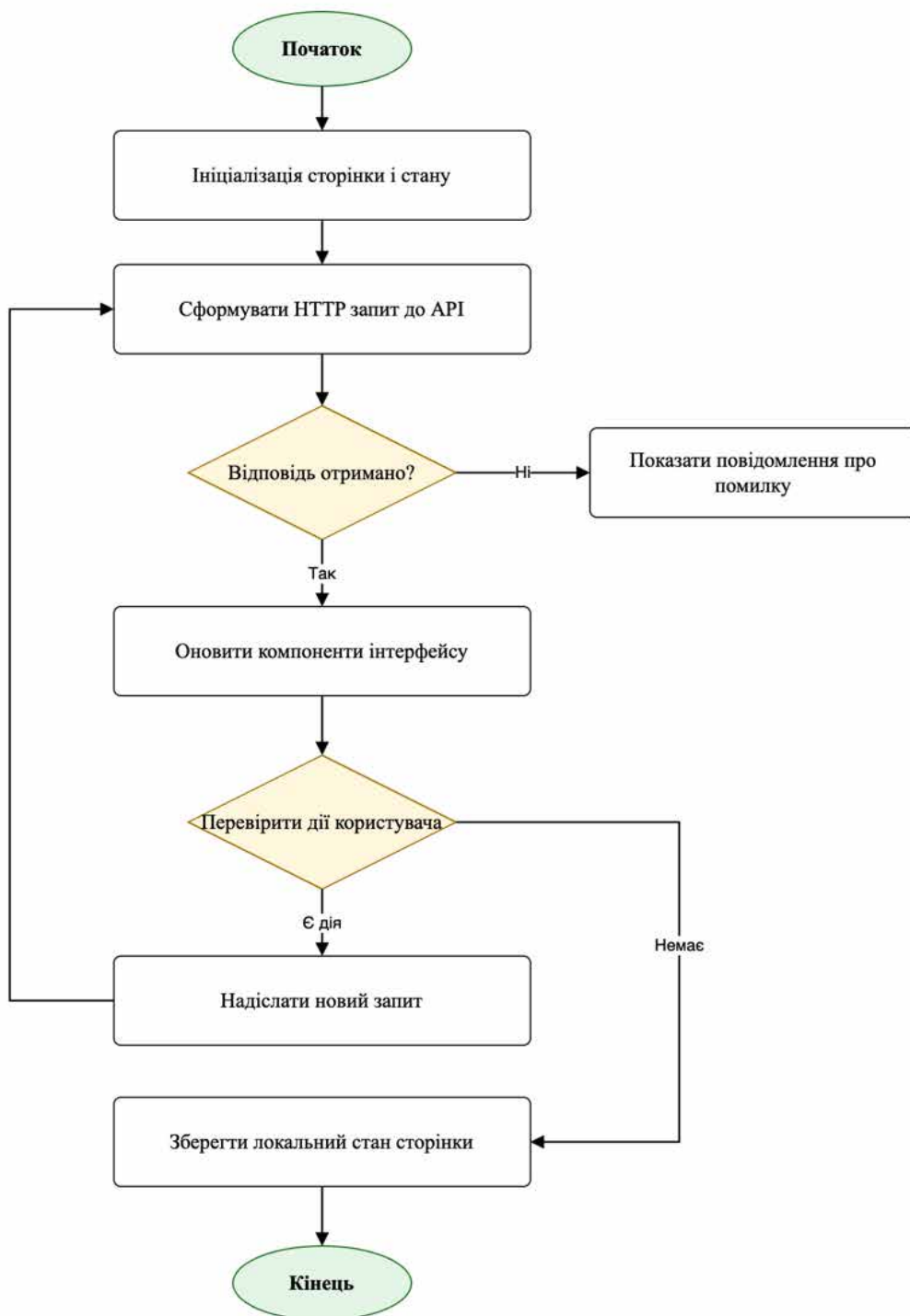


Рисунок 3.4 - Алгоритм роботи клієнтського модуля

Клієнтський модуль відповідає за ініціалізацію сторінки, формування HTTP-запиту до серверної частини, оброблення відповіді та оновлення елементів

інтерфейсу. Якщо відповідь від сервера не отримано або виникла помилка, користувачу показується відповідне повідомлення. Якщо відповідь успішна, інтерфейс оновлюється відповідно до отриманих даних. У разі подальших дій користувача формується новий запит до API, а стан сторінки зберігається локально.

На рисунку 3.5 наведено алгоритм оброблення запиту серверною частиною.

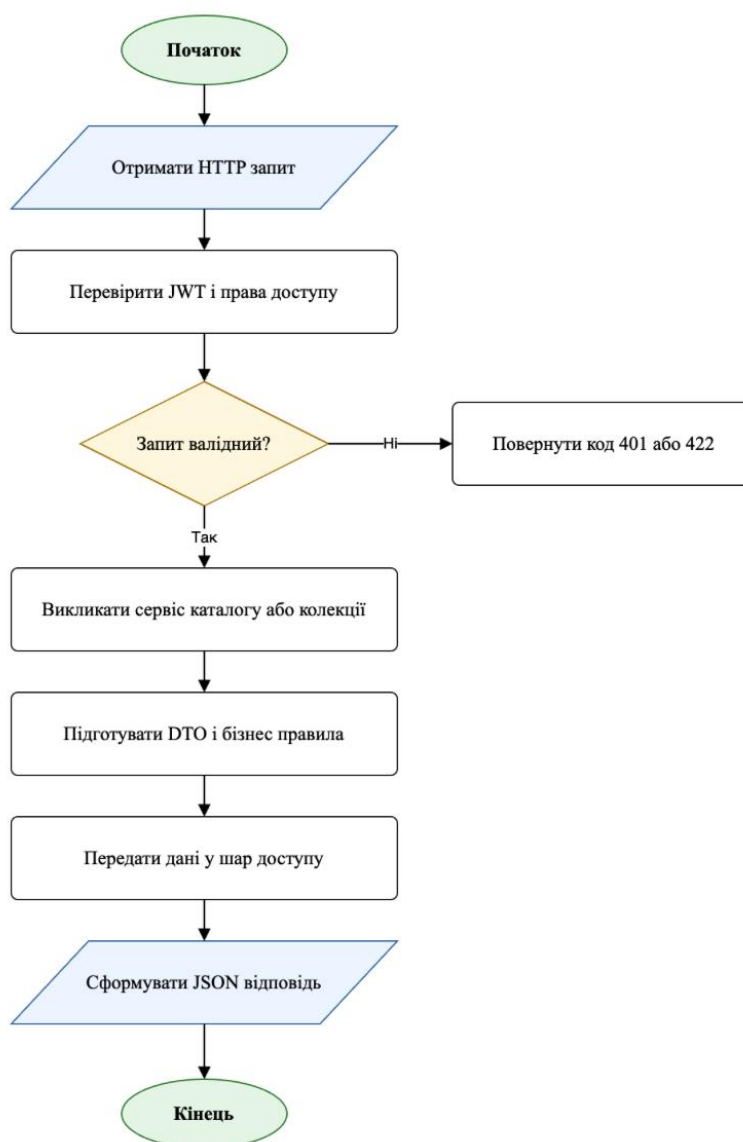


Рисунок 3.5 - Алгоритм оброблення HTTP-запиту серверним модулем

Серверний модуль приймає HTTP-запит, перевіряє авторизацію користувача та його права доступу. Якщо запит невалідний, система повертає код помилки 401 або 422. Якщо перевірку пройдено успішно, сервер викликає

відповідний сервіс каталогу, колекції або оцінювання. Після цього виконуються бізнес-правила, формується структура даних для збереження та запит передається до шару доступу до даних. Результатом роботи алгоритму є JSON-відповідь для клієнтської частини.

На рисунку 3.6 подано алгоритм збереження даних у базі.

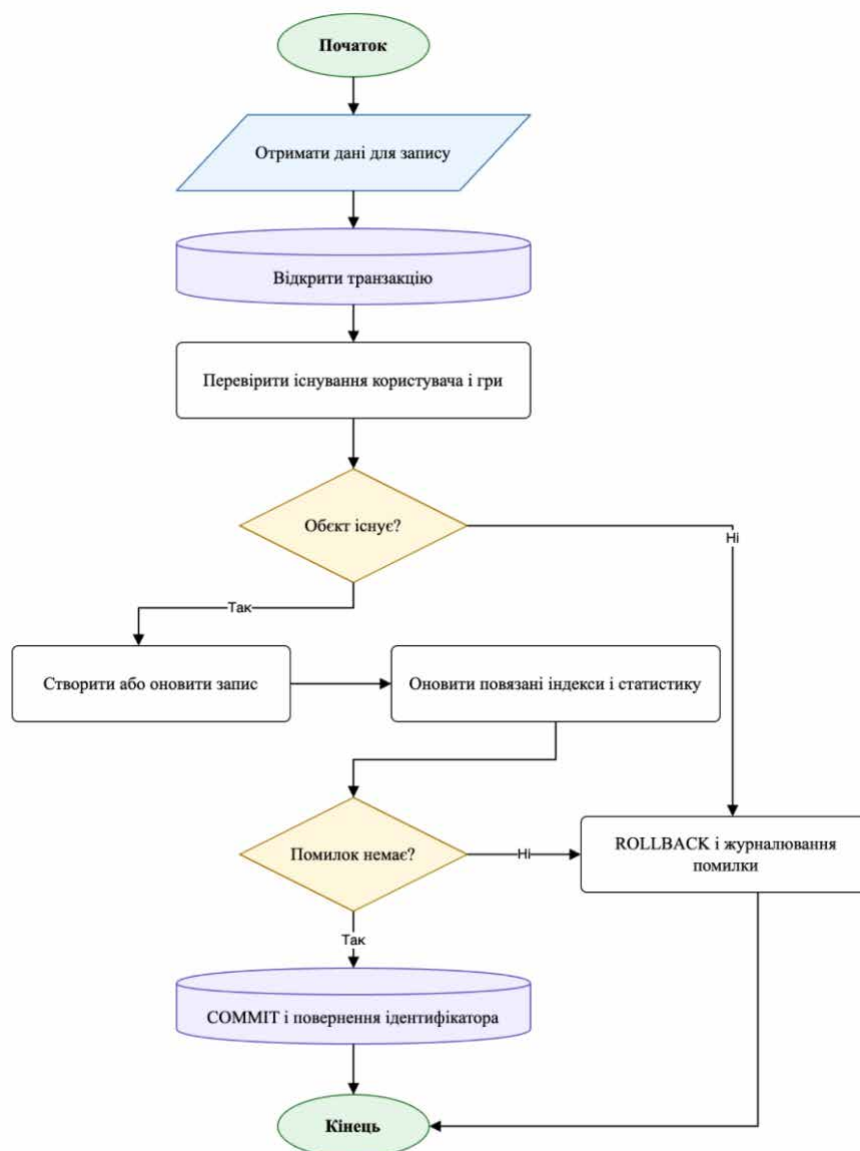


Рисунок 3.6 - Алгоритм збереження запису в базі даних

Алгоритм починається з отримання даних для запису. Далі відкривається транзакція та виконується перевірка існування користувача і гри. Якщо потрібні об'єкти відсутні, транзакція скасовується, а помилка фіксується в журналі. Якщо дані коректні, система створює або оновлює запис, після чого оновлює пов'язані індекси та статистичні показники. За відсутності помилок виконується COMMIT

і повертається ідентифікатор запису. Якщо виникає помилка, виконується ROLLBACK.

Таблиця 3.2 - Призначення основних алгоритмів системи

Алгоритм	Призначення	Основний результат
Додавання та оцінювання гри	Забезпечує додавання гри до колекції, введення оцінки та відгуку	Створений запис у колекції, збережена оцінка і відгук
Користувацький сценарій	Описує послідовність дій користувача в інтерфейсі	Виконання вибраної дії та повідомлення про результат
Клієнтський модуль	Формує запити до сервера та оновлює сторінку	Актуалізований інтерфейс користувача
Серверний модуль	Перевіряє запити, права доступу та викликає бізнес-логіку	JSON-відповідь або повідомлення про помилку
Модуль збереження даних	Забезпечує транзакційне створення або оновлення записів	Збережені дані з контролем цілісності

Реалізовані алгоритми узгоджуються зі структурою програмної системи та фізичною моделлю даних. Клієнтська частина відповідає за введення й відображення інформації, серверна частина виконує перевірку запитів і бізнес-логіку, а модуль збереження даних забезпечує цілісність операцій із користувачами, іграми, колекціями, оцінками та відгуками.

Отже, алгоритмізація програмних модулів дозволила формалізувати основні процеси роботи системи: взаємодію користувача з каталогом, додавання гри до колекції, оцінювання, оброблення запитів і збереження даних. Запропоновані алгоритми є основою для реалізації стабільної, контрольованої та зрозумілої логіки програмного забезпечення.

3.4 Висновки до третього розділу

У третьому розділі було виконано проектування архітектури програмної системи та розроблено алгоритмічне забезпечення для системи колекціонування та оцінювання відеоігор за рейтингом користувачів.

У пункті 3.1 обґрунтовано вибір технологічних засобів реалізації. Для створення програмного продукту обрано веборієнтований підхід, що забезпечує запуск системи через браузер і не потребує встановлення окремого клієнтського застосунку. Основою серверної частини визначено Node.js та Express.js, які дозволяють реалізувати маршрути, авторизацію, роботу з каталогом, колекціями, відгуками, модерацією та адміністративною панеллю. Для клієнтської частини використано HTML, CSS і JavaScript, що забезпечує створення зручного інтерфейсу з темною палітрою, фільтрами, картками ігор і формами взаємодії.

У пункті 3.2 розроблено фізичну модель даних системи. Визначено структуру таблиць для збереження користувачів, відеоігор, розробників, видавців, жанрів, платформ, колекцій, цифрових і фізичних копій, оцінок та відгуків. Для забезпечення цілісності даних передбачено первинні й зовнішні ключі, обмеження CHECK, UNIQUE-обмеження та індекси. Запропонована структура відповідає логічній ER-моделі та може бути реалізована в PostgreSQL або адаптована до локальної JSON-бази, використаної в програмному прототипі.

У пункті 3.3 виконано алгоритмізацію основних програмних модулів. Описано алгоритми додавання гри до колекції, оцінювання, роботи користувацького сценарію, функціонування клієнтського модуля, оброблення HTTP-запиту сервером і збереження даних. Алгоритми враховують перевірку авторизації, валідацію введених даних, контроль прав доступу, оброблення помилок, оновлення статистики та збереження результатів у базі даних.

Розроблена архітектура передбачає чіткий поділ системи на клієнтську частину, серверну логіку та рівень збереження даних. Такий підхід спрощує супровід програмного забезпечення, полегшує тестування окремих модулів і створює умови для подальшого розширення функціоналу. У майбутньому система може бути доповнена модулем рекомендацій, імпортом бібліотек з ігрових платформ, розширеною аналітикою та повноцінною інтеграцією з реляційною СУБД.

Отже, у третьому розділі було сформовано технічну основу реалізації програмної системи. Обрані технології, фізична модель даних і розроблені

алгоритми забезпечують практичну реалізацію функцій, визначених у попередніх розділах: ведення каталогу відеоігор, формування користувацьких колекцій, оцінювання, написання відгуків, модерацію контенту та перегляд рейтингової інформації.

4 ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ

4.1 Тестування програмних модулів системи колекціонування та оцінювання відеоігор

Тестування програмних модулів системи колекціонування та оцінювання відеоігор спрямоване на перевірку коректності роботи основних функцій, стабільності оброблення користувацьких дій, правильності збереження даних, роботи механізмів авторизації, розмежування доступу та формування рейтингової інформації. Перевірка охоплює модулі реєстрації, входу в систему, каталогу ігор, користувацької колекції, списку бажаного, оцінювання, відгуків, модерації, адміністративного керування та аналітики.

Методика тестування передбачає модульне, функціональне, інтеграційне та навантажувальне тестування. Модульне тестування використовується для перевірки окремих частин системи, функціональне - для оцінювання виконання основних сценаріїв користувача, інтеграційне - для перевірки взаємодії між клієнтською частиною, серверними маршрутами та базою даних, навантажувальне - для оцінювання стабільності роботи при великій кількості записів у каталозі та колекціях.

Загальний план тестування програмних модулів наведено в таблиці 4.1.

Таблиця 4.1

План тестування основних програмних модулів системи

№	Модуль / компонент	Тип тестування	Тестові дії	Очікуваний результат
1	Модуль реєстрації та входу	Модульне	Реєстрація користувача, вхід із правильними та неправильними даними	Коректна авторизація, повідомлення про помилки
2	Модуль ролей доступу	Функціональне	Перевірка доступу геймера, модератора та адміністратора	Користувач бачить лише дозволені функції

3	Каталог відеоігор	Функціональне	Перегляд списку ігор, відкриття картки гри, пошук за назвою	Коректне відображення каталогу та даних гри
---	-------------------	---------------	---	---

Продовження таблиці 4.1

4	Фільтрація та сортування	Функціональне	Фільтрація за жанром, платформою, рейтингом, статусом	Система повертає релевантні записи
5	Модуль колекції	Інтеграційне	Додавання гри до колекції, зміна статусу проходження, введення годин	Дані зберігаються та відображаються в особистій колекції
6	Список бажаного	Функціональне	Додавання гри до wishlist і видалення зі списку	Список бажаного оновлюється без дублювання записів
7	Модуль оцінювання	Функціональне	Виставлення оцінки від 1 до 10, повторне редагування оцінки	Оцінка зберігається, рейтинг гри оновлюється
8	Модуль відгуків	Функціональне	Створення, перегляд і редагування текстового відгуку	Відгук коректно зберігається та прив'язується до гри
9	Модуль скарг	Функціональне	Подання скарги на відгук	Скарга передається в панель модератора
10	Панель модератора	Інтеграційне	Перегляд скарг, приховування або залишення відгуку	Статус відгуку та скарги оновлюється
11	Адміністративна панель	Функціональне	Додавання, редагування та видалення гри	Каталог оновлюється відповідно до дій адміністратора
12	Модуль аналітики	Функціональне	Перегляд рейтингу, статистики жанрів і статусів	Формуються коректні статистичні показники
13	База даних	Інтеграційне	Запис, оновлення, видалення та читання даних	Дані не втрачаються, зв'язки між

				об'єктами зберігаються
14	Продуктивність системи	Навантажувальне	Робота з каталогом на 50-70 ігор і кількома користувачами	Система працює стабільно без критичних затримок

Оцінювання результатів тестування виконується за такими критеріями: правильність виконання функції, коректність збереження даних, відсутність дублювання записів, коректне відображення повідомлень, дотримання ролей доступу та стабільність роботи інтерфейсу. Для кожного модуля перевіряється не лише успішний сценарій, а й ситуації з некоректними даними, повторними діями користувача або відсутністю прав доступу.

Під час тестування модуля авторизації перевіряється створення нового облікового запису, вхід за email і паролем, вихід із системи та блокування доступу до закритих сторінок без авторизації. Окремо перевіряється, що геймер не має доступу до панелі адміністратора та модератора, а службові функції відкриваються лише відповідним ролям.

Для модуля каталогу перевіряється відображення списку відеоігор із реальними зображеннями, назвами, жанрами, платформами та рейтингами. Додатково тестуються пошук, фільтрація, сортування та відкриття детальної сторінки гри. Очікуваним результатом є правильне відображення даних без порушення структури сторінки та без втрати інформації після оновлення.

Модуль колекції тестується через додавання гри до особистої бібліотеки, зміну статусу проходження, вибір типу копії, введення кількості годин і додавання нотатки. Перевіряється, що одна й та сама гра не дублюється в колекції одного користувача, а зміни коректно зберігаються в базі даних і відображаються після повторного входу в систему.

Під час тестування модуля оцінювання перевіряється введення допустимих значень від 1 до 10, оновлення вже існуючої оцінки та перерахунок середнього рейтингу гри. Для модуля відгуків перевіряється збереження тексту, прив'язка відгуку до конкретного користувача та гри, а також відображення відгуків на сторінці гри.

Модуль модерації тестується через подання скарги на відгук і подальшу обробку цієї скарги модератором. Перевіряється, що скарга потрапляє до відповідної панелі, модератор може змінити статус контенту, а результат модерації відображається в системі. Це дозволяє контролювати якість користувацьких матеріалів і зменшити ризик появи некоректного контенту.

Адміністративний модуль перевіряється шляхом додавання нової гри до каталогу, редагування її характеристик і видалення запису. У процесі тестування контролюється правильність оновлення даних, збереження зв'язків із жанрами, платформами, розробниками та видавцями, а також недопущення некоректних або порожніх записів.

Інтеграційне тестування охоплює взаємодію між клієнтським інтерфейсом, серверними маршрутами Node.js / Express.js і локальною базою даних у форматі JSON. Перевіряється, чи правильно сервер приймає дані з форм, обробляє їх, зберігає в базі та повертає оновлену інформацію на сторінку користувача. Особлива увага приділяється операціям, де одночасно змінюються кілька пов'язаних об'єктів: колекція, оцінка, відгук і статистика гри.

Навантажувальне тестування проводиться на каталозі з розширеним набором відеоігор. Перевіряється швидкість відкриття головної сторінки, каталогу, сторінок фільтрації, колекції та аналітики. Також оцінюється стабільність роботи при послідовному додаванні великої кількості ігор до колекції, створенні відгуків і виконанні адміністративних дій.

У процесі тестування підтверджується, що розроблена система виконує основні функції предметної області: забезпечує реєстрацію й авторизацію користувачів, підтримує каталог відеоігор, дозволяє формувати персональні колекції, зберігає оцінки та відгуки, забезпечує модерацію контенту й формує рейтингову інформацію. Отримані результати тестування є основою для подальшого оцінювання ефективності системи та перевірки її готовності до практичного використання.

4.2 Тестування інтерфейсних модулів системи колекціонування та оцінювання відеоігор

Тестування інтерфейсних модулів виконано для перевірки зручності роботи користувача із системою, коректності відображення сторінок, правильності переходів між розділами та відповідності інтерфейсу функціональним вимогам. Основну увагу приділено головній сторінці, формам авторизації та реєстрації, каталогу відеоігор, сторінці гри, модулю колекції та аналітичній панелі.

На рисунку 4.1 подано головну сторінку системи. Вона містить короткий опис призначення програмного продукту, навігаційні елементи, кнопки переходу до каталогу та реєстрації, а також узагальнені показники: кількість ігор у каталозі, відгуків, записів колекцій та користувачів. Під час тестування перевірено коректність переходів із головної сторінки до каталогу, входу та реєстрації.

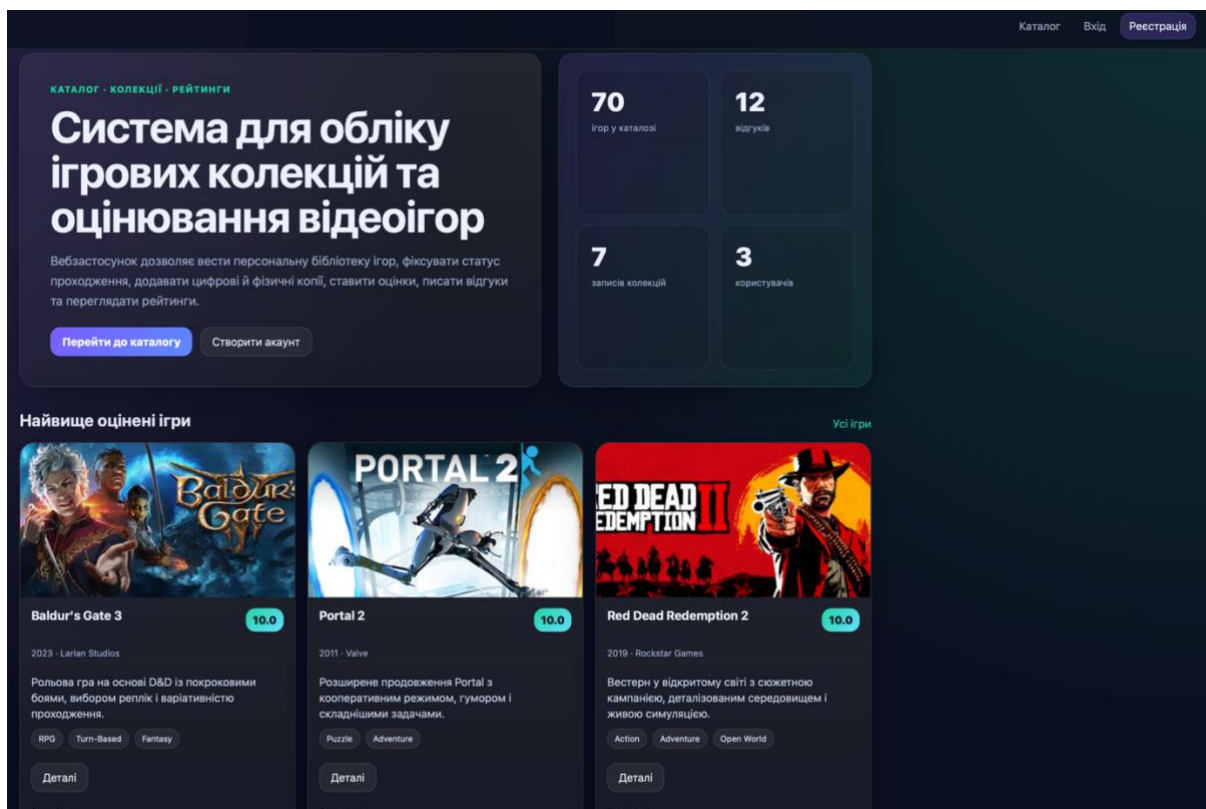
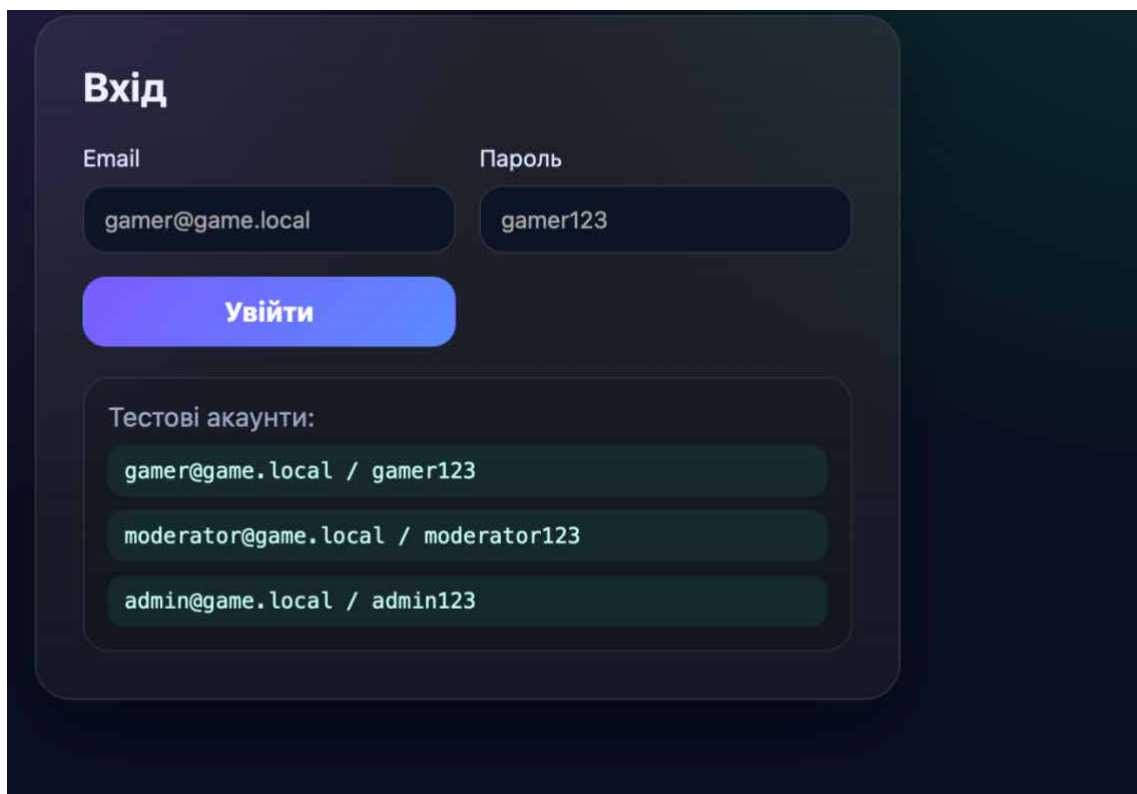


Рисунок 4.1 - Головна сторінка системи колекціонування та оцінювання відеоігор

На рисунку 4.2 наведено сторінку входу до системи. Перевірено введення тестових облікових даних, оброблення неправильного пароля, успішний вхід користувача та перенаправлення до каталогу. Також перевірено доступність тестових акаунтів для ролей геймера, модератора та адміністратора.



The image shows a login interface with a dark blue background. At the top left, the word "Вхід" (Login) is displayed in white. Below it, there are two input fields: "Email" and "Пароль" (Password). The "Email" field contains the text "gamer@game.local" and the "Password" field contains "gamer123". Below these fields is a blue button with the text "Увійти" (Login). Underneath the button, there is a section titled "Тестові акаунти:" (Test accounts:). This section contains three lines of text, each representing a test account: "gamer@game.local / gamer123", "moderator@game.local / moderator123", and "admin@game.local / admin123".

Рисунок 4.2 - Інтерфейс входу до системи

На рисунку 4.3 подано форму реєстрації нового користувача. Під час тестування перевірено введення імені, email і пароля, обмеження мінімальної довжини пароля, створення нового облікового запису та неможливість повторної реєстрації з уже використаною електронною поштою.

Реєстрація

Ім'я

Email

Пароль

Створити акаунт

Вже є акаунт? [Увійти](#)

Рисунок 4.3 - Інтерфейс реєстрації користувача

На рисунку 4.4 показано сторінку каталогу відеоігор. У каталозі відображається 70 ігор із зображеннями, назвами, роком випуску, розробниками, коротким описом, жанрами та рейтингами. Перевірено роботу пошуку, фільтрації за жанром і платформою, сортування за рейтингом, кнопки перегляду деталей, додавання гри до списку бажаного та відображення статусу гри в колекції.

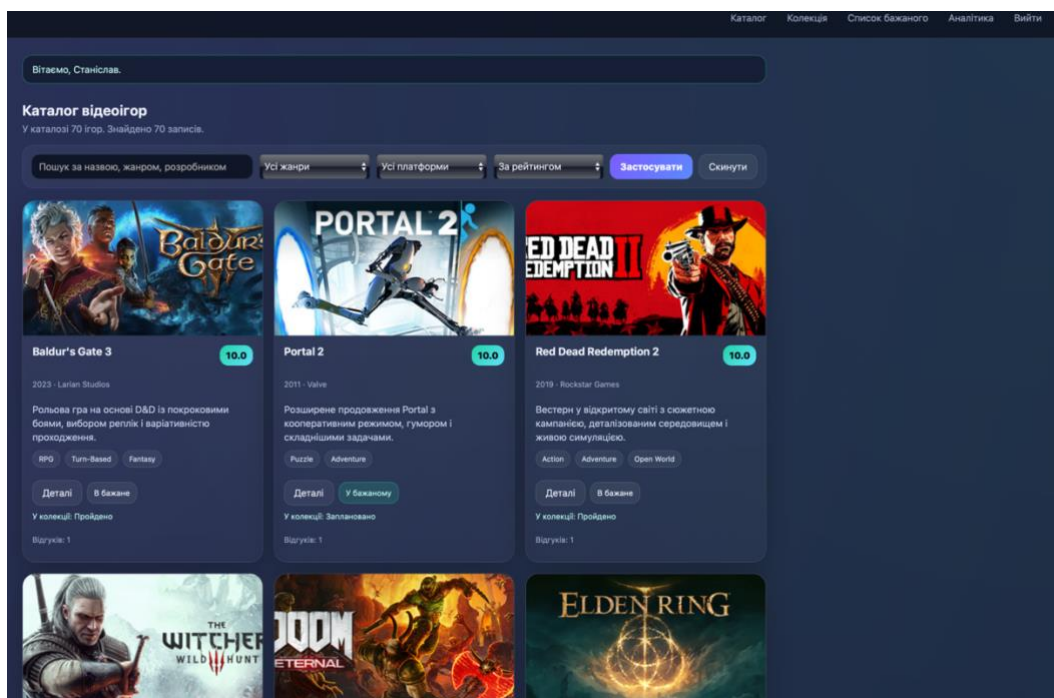


Рисунок 4.4 - Інтерфейс каталогу відеоігор

На рисунку 4.5 наведено сторінку конкретної гри. Вона містить основну інформацію про гру, рейтинг, жанри, платформи, форму оновлення запису колекції та форму оцінювання. Під час тестування перевірено зміну статусу проходження, введення кількості годин, вибір типу копії, збереження примітки, виставлення оцінки та написання відгуку. Також перевірено, що після збереження дані відображаються у блоці відгуків користувачів.

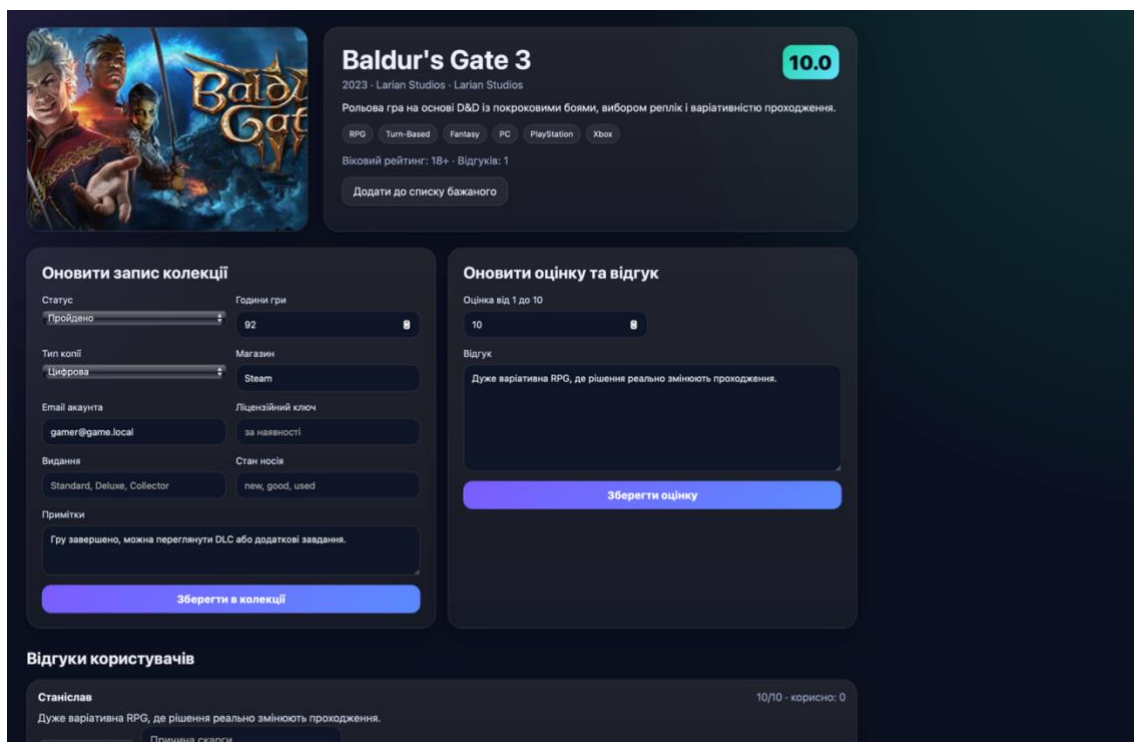


Рисунок 4.5 - Сторінка гри з формами колекції, оцінки та відгуку

На рисунку 4.6 подано сторінку особистої колекції користувача. У цьому модулі перевірено відображення всіх доданих ігор, правильність статусів проходження, типів копій, кількості зіграних годин і приміток. Окремо перевірено видалення гри з колекції та оновлення підсумкових показників: загальної кількості ігор і сумарного часу проходження.

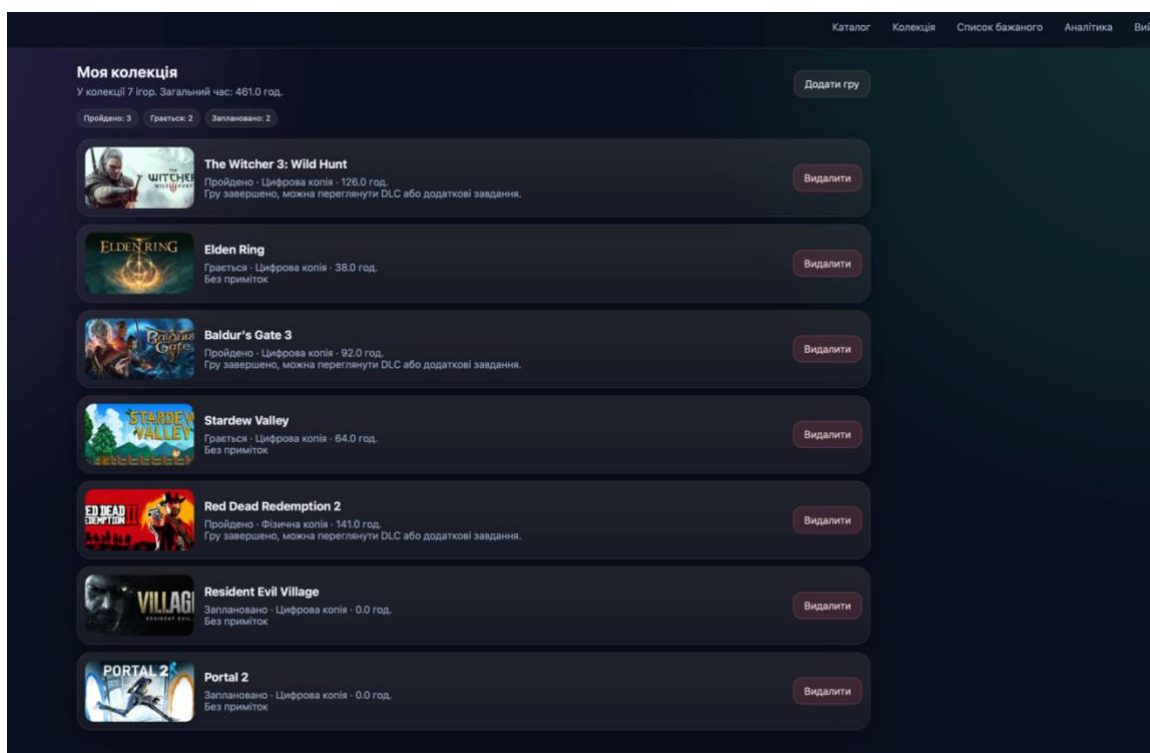


Рисунок 4.6 - Інтерфейс особистої ігрової колекції користувача

На рисунку 4.7 наведено аналітичну панель системи. Вона відображає топ ігор за рейтингом, кількість ігор за жанрами, статуси колекцій та активність користувачів. Під час тестування перевірено правильність підрахунку середнього рейтингу, кількості записів за жанрами, статусів проходження та активності користувачів.

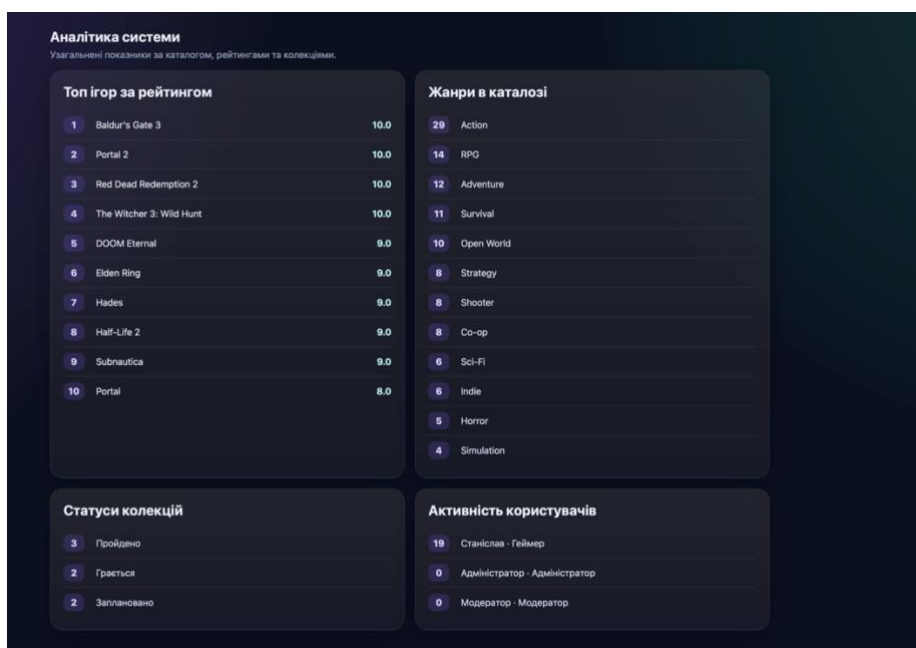


Рисунок 4.7 - Інтерфейс аналітичної панелі системи

Таблиця 4.2

Результати тестування інтерфейсних модулів

№	Інтерфейсний модуль	Перевірені дії	Результат
1	Головна сторінка	Перехід до каталогу, входу та реєстрації, перегляд статистики	Переходи працюють, статистичні блоки відображаються коректно
2	Форма входу	Вхід за тестовими акаунтами, перевірка неправильних даних	Авторизація працює, помилки обробляються
3	Форма реєстрації	Створення нового акаунта, перевірка email і пароля	Дані зберігаються, дублювання email не допускається
4	Каталог відеоігор	Пошук, фільтрація, сортування, відкриття картки гри	Каталог працює стабільно, ігри відображаються з реальними зображеннями
5	Сторінка гри	Оновлення колекції, введення годин, оцінка, відгук	Дані зберігаються та відображаються після оновлення сторінки
6	Моя колекція	Перегляд доданих ігор, підрахунок годин, видалення запису	Колекція оновлюється без втрати даних
7	Аналітика	Перегляд рейтингів, жанрів, статусів і активності	Показники формуються коректно на основі даних системи

За результатами тестування встановлено, що інтерфейсні модулі відповідають основним вимогам системи. Навігація між сторінками працює коректно, форми мають зрозумілу структуру, дані після введення зберігаються та відображаються у відповідних розділах. Темна палітра інтерфейсу забезпечує цілісний візуальний стиль, а карткова структура каталогу спрощує перегляд великої кількості ігор.

Окремо підтверджено правильність взаємодії інтерфейсу з функціональними модулями системи. Додавання гри до колекції змінює її статус у каталозі, оцінювання оновлює рейтинг гри, відгук з'являється на сторінці гри, а аналітична панель відображає актуальні узагальнені показники. Це свідчить про узгоджену роботу клієнтської частини, серверної логіки та бази даних.

Отже, тестування інтерфейсних модулів підтвердило працездатність основних сценаріїв користувача: реєстрації, входу, перегляду каталогу,

формування колекції, оцінювання відеоігор, написання відгуків і перегляду аналітики. Отримані результати показують, що інтерфейс системи є функціональним, зрозумілим і придатним для практичного використання.

4.3 Результати тестування та аналіз ефективності системи, склад інсталяційного пакету

За результатами тестування встановлено, що розроблена система колекціонування та оцінювання відеоігор коректно виконує основні функції, визначені у попередніх розділах. Перевірено роботу реєстрації та авторизації користувачів, перегляд каталогу, пошук і фільтрацію ігор, додавання записів до колекції, ведення списку бажаного, оновлення статусів проходження, збереження оцінок, відгуків і формування аналітичних показників.

Під час тестування використовувався каталог із 70 відеоігор, що дозволило перевірити роботу системи не лише на мінімальному наборі даних, а й на розширеній базі записів. У процесі перевірки підтверджено, що сторінки каталогу відкриваються стабільно, фільтри за жанром і платформою працюють коректно, а сортування за рейтингом відображає ігри у правильному порядку.

Результати тестування основних функцій системи наведено в таблиці 4.3.

Таблиця 4.3

Результати функціонального тестування системи

№	Перевірена функція	Результат виконання	Стан
1	Реєстрація нового користувача	Обліковий запис створюється, дані зберігаються	Успішно
2	Авторизація користувача	Вхід виконується за правильними обліковими даними	Успішно
3	Розмежування ролей	Геймер, модератор і адміністратор мають різні права доступу	Успішно
4	Перегляд каталогу ігор	Відображаються картки ігор із зображеннями, жанрами та рейтингами	Успішно
5	Пошук і фільтрація	Система повертає записи відповідно до введених параметрів	Успішно
6	Додавання гри до колекції	Запис створюється та відображається в особистій колекції	Успішно

7	Оновлення статусу проходження	Статус, години гри та примітки зберігаються коректно	Успішно
8	Додавання до списку бажаного	Гра відображається у відповідному розділі користувача	Успішно

Продовження таблиці 4.3

9	Оцінювання гри	Оцінка зберігається, середній рейтинг оновлюється	Успішно
10	Написання відгуку	Відгук прив'язується до користувача та конкретної гри	Успішно
11	Подання скарги	Скарга передається до модуля модерації	Успішно
12	Аналітика системи	Формуються рейтинги, статистика жанрів і статусів колекцій	Успішно

Оцінювання ефективності системи виконувалося за критеріями швидкодії, стабільності, зручності використання та коректності роботи з даними. Під час виконання типових операцій, таких як відкриття каталогу, перегляд картки гри, додавання запису до колекції та збереження оцінки, критичних затримок не виявлено. Система стабільно обробляє послідовні запити користувача та зберігає внесені зміни після оновлення сторінки.

Під час перевірки роботи з даними встановлено, що система не допускає дублювання однакових записів у колекції одного користувача, коректно оновлює вже наявні оцінки та забезпечує прив'язку відгуку до конкретної гри. Аналітичний модуль правильно обчислює рейтинги ігор, кількість записів за жанрами, статуси колекцій і активність користувачів.

Основні показники ефективності наведено в таблиці 4.4.

Таблиця 4.4

Оцінювання ефективності роботи системи

Критерій	Результат перевірки	Оцінка
Швидкість відкриття головної сторінки	Сторінка відкривається без помітної затримки	Достатня
Робота каталогу з 70 іграми	Картки ігор завантажуються коректно	Достатня
Пошук і фільтрація	Результати відповідають заданим параметрам	Коректно
Збереження колекції	Дані не втрачаються після повторного входу	Коректно
Оновлення рейтингу	Середній бал змінюється після оцінювання	Коректно

Розмежування доступу	Службові функції недоступні звичайному користувачу	Коректно
Стабільність роботи	Критичних помилок під час основних сценаріїв не виявлено	Стабільно

Для розгортання системи передбачено клієнт-серверну архітектуру. Її загальну структуру наведено на рисунку 4.8.

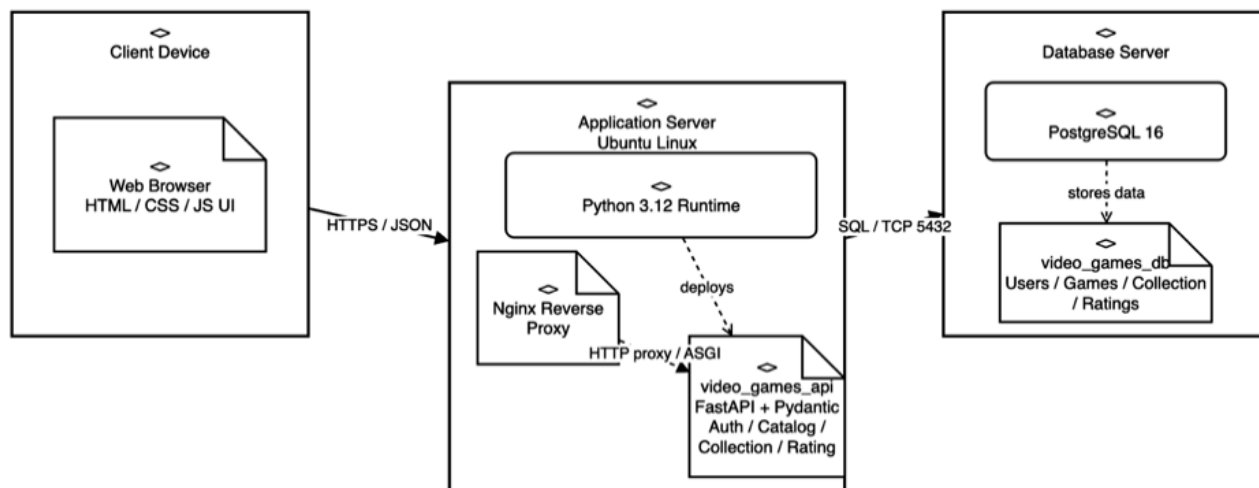


Рисунок 4.8 - Діаграма розгортання системи колекціонування та оцінювання відеоігор

Згідно з діаграмою, користувач працює із системою через веббраузер на клієнтському пристрої. Передавання даних між клієнтом і сервером виконується через HTTPS/JSON. Сервер застосунку розміщується на Linux-середовищі та містить вебсервер, прикладний runtime і серверний застосунок. Для приймання зовнішніх запитів може використовуватися Nginx Reverse Proxy, який передає запити до прикладного API. Дані системи зберігаються на окремому сервері бази даних PostgreSQL, де розміщуються записи про користувачів, ігри, колекції, оцінки та відгуки.

У навчальній реалізації система може запускатися локально на одному комп'ютері. Такий варіант спрощує перевірку програмного продукту, демонстрацію функціоналу та тестування основних сценаріїв без додаткового налаштування серверної інфраструктури.

Склад інсталяційного пакету наведено в таблиці 4.5.

Таблиця 4.5

Склад інсталяційного пакету системи

Елемент пакету	Призначення
package.json	Опис проєкту, залежностей і команд запуску
server.js або app.js	Основний файл запуску серверної частини
routes	Маршрути для каталогу, авторизації, колекції, відгуків, модерації та адміністративної панелі
views або templates	Шаблони сторінок інтерфейсу користувача
public	Статичні файли: CSS, JavaScript, зображення та елементи оформлення
data	Файли локальної бази даних або початкові дані системи
database.json	Дані про користувачів, ігри, колекції, рейтинги та відгуки у навчальному прототипі
run.bat	Файл швидкого запуску системи у Windows
run.command	Файл швидкого запуску системи у macOS
README.md	Коротка інструкція зі встановлення, запуску та тестових облікових записів

Для запуску системи користувач розпаковує інсталяційний пакет, встановлює залежності та запускає сервер. Після цього застосунок відкривається у браузері за локальною адресою. Для демонстрації роботи передбачено тестові облікові записи геймера, модератора та адміністратора, що дозволяє перевірити різні рівні доступу без додаткового налаштування.

У результаті тестування підтверджено, що система виконує основні функції, необхідні для обліку ігрових колекцій та оцінювання відеоігор. Розроблений програмний продукт забезпечує зручний доступ до каталогу, підтримує персональні колекції, дозволяє залишати оцінки й відгуки, формує рейтинги та надає аналітичну інформацію. Це підтверджує працездатність системи та доцільність використання запропонованих архітектурних і програмних рішень.

4.4 Висновки до четвертого розділу

У четвертому розділі було проведено тестування та оцінювання ефективності системи колекціонування та оцінювання відеоігор за рейтингом користувачів. Перевірка охопила програмні модулі, інтерфейсні компоненти, роботу з даними, механізми авторизації, розмежування ролей, формування колекцій, оцінювання, відгуки, модерацію та аналітичні функції.

У пункті 4.1 було сформовано план тестування основних програмних модулів системи. Перевірено модулі реєстрації та входу, каталогу відеоігор, пошуку, фільтрації, керування колекцією, списку бажаного, оцінювання, відгуків, скарг, модерації, адміністративного керування та аналітики. У процесі тестування підтверджено, що система коректно обробляє дії користувачів, зберігає дані та не допускає некоректного доступу до службових функцій.

У пункті 4.2 було виконано тестування інтерфейсних модулів. Перевірено головну сторінку, форму входу, форму реєстрації, каталог відеоігор, сторінку конкретної гри, особисту колекцію та аналітичну панель. Установлено, що інтерфейс забезпечує зручну навігацію, коректне відображення карток ігор, реальних зображень, рейтингів, статусів проходження, форм введення оцінок і відгуків. Дані після виконання користувацьких дій зберігаються та відображаються у відповідних розділах системи.

У пункті 4.3 було узагальнено результати тестування та виконано аналіз ефективності системи. Під час перевірки використовувався каталог із 70 відеоігор, що дозволило оцінити роботу системи на розширеному наборі даних. Підтверджено стабільність роботи пошуку, фільтрації, сортування, додавання ігор до колекції, оновлення статусів, збереження оцінок і формування аналітичних показників. Критичних помилок під час виконання основних сценаріїв не виявлено.

Окремо було визначено склад інсталяційного пакету системи. До нього входять файли серверної частини, маршрути, шаблони сторінок, статичні ресурси, локальна база даних, файли швидкого запуску та інструкція

користувача. Такий склад пакету забезпечує можливість швидкого розгортання програмного продукту на локальному комп'ютері та демонстрації основних функцій без складного налаштування інфраструктури.

Отже, результати тестування підтвердили працездатність розробленої системи та відповідність її основним функціональним вимогам. Система забезпечує реєстрацію й авторизацію користувачів, ведення каталогу відеоігор, формування персональних колекцій, роботу зі списком бажаного, оцінювання, написання відгуків, модерацію контенту та перегляд аналітики. Проведене тестування показало, що розроблений програмний продукт є стабільним, зручним у використанні та придатним для практичного застосування як система обліку ігрових колекцій і формування користувацького рейтингу відеоігор.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було спроектовано та розроблено програмне забезпечення з колекціонування та оцінювання відеоігор за рейтингом користувачів. Розроблена система призначена для централізованого ведення каталогу відеоігор, формування персональних ігрових колекцій, збереження статусів проходження, оцінок, відгуків та перегляду аналітичної інформації.

У першому розділі було досліджено предметну область колекціонування та оцінювання відеоігор. Визначено, що користувачам потрібен зручний інструмент для обліку цифрових і фізичних копій ігор, контролю проходження, збереження власних вражень і перегляду рейтингів. Проведено аналіз існуючих рішень, зокрема Backloggd, Grouvee, Playnite, RAWG та IGDB. Установлено, що ці сервіси частково вирішують окремі задачі, однак не забезпечують повного поєднання персональної колекції, оцінювання, відгуків, модерації, аналітики та власного керування даними в межах однієї системи.

У процесі моделювання предметної області було визначено основних учасників системи: геймера, модератора та адміністратора. Побудовано DFD-діаграму, діаграму прецедентів, діаграму послідовності та діаграму активності, які описують основні процеси роботи системи. На основі цих моделей сформовано функціональні, нефункціональні та безпекові вимоги до програмного забезпечення.

У другому розділі було виконано проєктування інформаційного та програмного забезпечення. Розроблено логічну ER-модель даних, яка охоплює користувачів, відеоігри, розробників, видавців, жанри, платформи, колекції, цифрові та фізичні копії, оцінки й відгуки. Модель приведено до третьої нормальної форми, що дозволило зменшити дублювання даних і забезпечити цілісність зв'язків між сутностями. Також було побудовано діаграму класів,

кооперації, діаграму компонентів і діаграму пакетів, які визначили структуру програмної системи та взаємодію її основних частин.

У третьому розділі було обґрунтовано вибір технологічних засобів реалізації. Для створення системи використано веборієнтований підхід, Node.js, Express.js, HTML, CSS, JavaScript та локальну базу даних у форматі JSON. Такий стек забезпечив простий запуск, зручне тестування та достатню функціональність для навчального програмного продукту. Розроблено фізичну модель даних, SQL-структуру таблиць, обмеження цілісності та індекси. Також описано алгоритми роботи основних програмних модулів: додавання гри до колекції, оцінювання, оброблення клієнтських і серверних запитів та збереження даних.

Практичним результатом роботи став вебзастосунок із темним інтерфейсом, каталогом із 70 відеоігор, реєстрацією та авторизацією користувачів, розмежуванням ролей, особистою колекцією, списком бажаного, оцінками, відгуками, скаргами, панеллю модератора, адміністративним керуванням каталогом та аналітичною панеллю. Система дозволяє користувачеві переглядати каталог, фільтрувати ігри, додавати їх до колекції, вказувати статус проходження, тип копії, кількість годин, залишати оцінки та текстові відгуки.

У четвертому розділі було проведено тестування програмних та інтерфейсних модулів системи. Перевірено роботу реєстрації, входу, каталогу, пошуку, фільтрації, колекції, wishlist, оцінювання, відгуків, модерації, адміністративної панелі та аналітики. Результати тестування показали, що система коректно обробляє основні сценарії користувача, зберігає дані, не допускає некоректного доступу до службових функцій і стабільно працює з розширеним набором ігор.

Отже, мету кваліфікаційної роботи досягнуто. Було створено програмне забезпечення, яке забезпечує централізований облік ігрових колекцій, підтримує оцінювання відеоігор, роботу з відгуками, модерацію контенту та формування рейтингової інформації. Розроблена система є функціонально завершеним прототипом, придатним для практичного використання, демонстрації та

подальшого розвитку. У майбутньому її можна розширити за рахунок інтеграції з зовнішніми ігровими платформами, імпорту бібліотек користувача, рекомендаційної системи, повноцінної PostgreSQL-бази даних і розширеної статистики ігрової активності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Node.js Documentation. Node.js: офіційна документація.
URL: <https://nodejs.org/docs/latest/api/>
2. Node.js. About Node.js. URL: <https://nodejs.org/en/about>
3. Express.js. Node.js web application framework: офіційний сайт.
URL: <https://expressjs.com/en/>
4. Express.js. GitHub repository. URL: <https://github.com/expressjs/express>
5. MDN Web Docs. HTML: HyperText Markup Language.
URL: <https://developer.mozilla.org/en-US/docs/Web/HTML>
6. MDN Web Docs. CSS: Cascading Style Sheets.
URL: <https://developer.mozilla.org/en-US/docs/Web/CSS>
7. MDN Web Docs. JavaScript. URL: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
8. MDN Web Docs. HTTP resources and specifications.
URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Resources_and_specifications
9. npm Docs. Офіційна документація npm. URL: <https://docs.npmjs.com/>
10. npm Docs. package.json.
URL: <https://docs.npmjs.com/files/package.json/>
11. PostgreSQL Global Development Group. PostgreSQL Documentation.
URL: <https://www.postgresql.org/docs/>
12. PostgreSQL Global Development Group. PostgreSQL 16 Documentation.
URL: <https://www.postgresql.org/files/documentation/pdf/16/postgresql-16-A4.pdf>
13. Ecma International. ECMA-404: The JSON Data Interchange Syntax.
URL: <https://ecma-international.org/publications-and-standards/standards/ecma-404/>
14. Bray T. The JavaScript Object Notation (JSON) Data Interchange Format. RFC 7159. URL: <https://datatracker.ietf.org/doc/html/rfc7159>

15. Fielding R., Nottingham M., Reschke J. HTTP Semantics. RFC 9110.
URL: <https://datatracker.ietf.org/doc/html/rfc9110>
16. Object Management Group. Unified Modeling Language Specification, version 2.5.1. URL: <https://www.omg.org/spec/UML/2.5.1/About-UML>
17. OWASP Foundation. Application Security Verification Standard.
URL: <https://owasp.org/www-project-application-security-verification-standard/>
18. OWASP Foundation. OWASP Top 10:2021.
URL: <https://owasp.org/Top10/2021/>
19. NGINX Documentation. NGINX Reverse Proxy.
URL: <https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/>
20. Backloggd. A Video Game Collection Tracker.
URL: <https://backloggd.com/>
21. Grouvee. Track Your Video Game Backlog, Rate and Review Games.
URL: <https://www.grouvee.com/>
22. Playnite. Video game library manager. URL: <https://playnite.link/>
23. Playnite. GitHub repository.
URL: <https://github.com/JosefNemec/Playnite>
24. RAWG. The Biggest Video Game Database. URL: <https://rawg.io/>
25. IGDB. Discover, Rate and Track Your Games.
URL: <https://www.igdb.com/>
26. IGDB. Video Game Database API. URL: <https://www.igdb.com/api>
27. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Boston: Addison-Wesley, 2004. 208 p.
28. Sommerville I. Software Engineering. 10th ed. Boston: Pearson, 2015. 816 p.
29. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. 9th ed. New York: McGraw-Hill Education, 2019. 704 p.
30. Connolly T., Begg C. Database Systems: A Practical Approach to Design, Implementation, and Management. 6th ed. Harlow: Pearson, 2015. 1440 p.

ДОДАТОК А

Лістинг серверної частини програмного забезпечення

```
const express = require("express");
const session = require("express-session");
const fs = require("fs");
const path = require("path");
const bcrypt = require("bcryptjs");

const app = express();
const PORT = 3000;
const DB_PATH = path.join(__dirname, "data", "db.json");

app.set("view engine", "ejs");
app.set("views", path.join(__dirname, "views"));

app.use(express.urlencoded({ extended: true }));
app.use(express.json());
app.use(express.static(path.join(__dirname, "public")));

app.use(
  session({
    secret: "game-collection-secret",
    resave: false,
    saveUninitialized: false
  })
);

function readDB() {
  const raw = fs.readFileSync(DB_PATH, "utf8");
  return JSON.parse(raw);
}

function writeDB(data) {
  fs.writeFileSync(DB_PATH, JSON.stringify(data, null, 2), "utf8");
}

function requireAuth(req, res, next) {
  if (!req.session.user) {
    return res.redirect("/login");
  }
  next();
}

function requireRole(role) {
  return function (req, res, next) {
    if (!req.session.user || req.session.user.role !== role) {
      return res.status(403).send("Недостатньо прав доступу");
    }
    next();
  };
}

function getGameRating(gameId, reviews) {
  const gameReviews = reviews.filter((review) => review.gameId ===
    Number(gameId));

  if (gameReviews.length === 0) {
    return {
      average: 0,
      count: 0
    };
  }
}
```

```

    }

    const sum = gameReviews.reduce((acc, review) => acc + Number(review.rating),
0);

    return {
      average: Number((sum / gameReviews.length).toFixed(1)),
      count: gameReviews.length
    };
  }
}

app.get("/", (req, res) => {
  const db = readDB();

  res.render("index", {
    user: req.session.user,
    gamesCount: db.games.length,
    reviewsCount: db.reviews.length,
    collectionsCount: db.collections.length,
    usersCount: db.users.length
  });
});

app.get("/register", (req, res) => {
  res.render("register", { error: null });
});

app.post("/register", async (req, res) => {
  const db = readDB();
  const { username, email, password } = req.body;

  const exists = db.users.find((user) => user.email === email);

  if (exists) {
    return res.render("register", {
      error: "Користувач з таким email уже існує"
    });
  }

  const passwordHash = await bcrypt.hash(password, 10);

  const newUser = {
    id: Date.now(),
    username,
    email,
    passwordHash,
    role: "gamer",
    status: "active",
    createdAt: new Date().toISOString()
  };

  db.users.push(newUser);
  writeDB(db);

  req.session.user = {
    id: newUser.id,
    username: newUser.username,
    role: newUser.role
  };

  res.redirect("/catalog");
});

```

```

app.get("/login", (req, res) => {
  res.render("login", { error: null });
});

app.post("/login", async (req, res) => {
  const db = readDB();
  const { email, password } = req.body;

  const user = db.users.find((item) => item.email === email);

  if (!user) {
    return res.render("login", {
      error: "Неправильний email або пароль"
    });
  }

  const isValid = await bcrypt.compare(password, user.passwordHash);

  if (!isValid) {
    return res.render("login", {
      error: "Неправильний email або пароль"
    });
  }

  req.session.user = {
    id: user.id,
    username: user.username,
    role: user.role
  };

  res.redirect("/catalog");
});

app.get("/logout", (req, res) => {
  req.session.destroy(() => {
    res.redirect("/");
  });
});

app.get("/catalog", (req, res) => {
  const db = readDB();
  const { genre, platform, search } = req.query;

  let games = db.games.map((game) => {
    const rating = getGameRating(game.id, db.reviews);

    return {
      ...game,
      rating: rating.average,
      reviewsCount: rating.count
    };
  });

  if (genre) {
    games = games.filter((game) => game.genres.includes(genre));
  }

  if (platform) {
    games = games.filter((game) => game.platforms.includes(platform));
  }

  if (search) {
    games = games.filter((game) =>

```

```

        game.title.toLowerCase().includes(search.toLowerCase())
    );
}

res.render("catalog", {
    user: req.session.user,
    games,
    genres: db.genres,
    platforms: db.platforms,
    filters: { genre, platform, search }
});
});

app.get("/games/:id", (req, res) => {
    const db = readDB();
    const game = db.games.find((item) => item.id === Number(req.params.id));

    if (!game) {
        return res.status(404).send("Гру не знайдено");
    }

    const rating = getGameRating(game.id, db.reviews);
    const reviews = db.reviews.filter((review) => review.gameId === game.id);

    res.render("game", {
        user: req.session.user,
        game,
        rating,
        reviews
    });
});

app.post("/collection/add", requireAuth, (req, res) => {
    const db = readDB();
    const { gameId, copyType, status, hoursPlayed, note } = req.body;

    const exists = db.collections.find(
        (item) =>
            item.userId === req.session.user.id &&
            item.gameId === Number(gameId)
    );

    if (!exists) {
        db.collections.push({
            id: Date.now(),
            userId: req.session.user.id,
            gameId: Number(gameId),
            copyType,
            status,
            hoursPlayed: Number(hoursPlayed || 0),
            note: note || "",
            addedAt: new Date().toISOString()
        });

        writeDB(db);
    }

    res.redirect("/collection");
});

app.get("/collection", requireAuth, (req, res) => {
    const db = readDB();

```

```

const collection = db.collections
  .filter((item) => item.userId === req.session.user.id)
  .map((item) => {
    const game = db.games.find((g) => g.id === item.gameId);

    return {
      ...item,
      game
    };
  });

res.render("collection", {
  user: req.session.user,
  collection
});
});

app.post("/wishlist/add", requireAuth, (req, res) => {
  const db = readDB();
  const gameId = Number(req.body.gameId);

  const exists = db.wishlist.find(
    (item) =>
      item.userId === req.session.user.id &&
      item.gameId === gameId
  );

  if (!exists) {
    db.wishlist.push({
      id: Date.now(),
      userId: req.session.user.id,
      gameId,
      addedAt: new Date().toISOString()
    });

    writeDB(db);
  }

  res.redirect(`/games/${gameId}`);
});

app.post("/reviews/add", requireAuth, (req, res) => {
  const db = readDB();
  const { gameId, rating, text } = req.body;

  const oldReview = db.reviews.find(
    (review) =>
      review.userId === req.session.user.id &&
      review.gameId === Number(gameId)
  );

  if (oldReview) {
    oldReview.rating = Number(rating);
    oldReview.text = text;
    oldReview.updatedAt = new Date().toISOString();
  } else {
    db.reviews.push({
      id: Date.now(),
      userId: req.session.user.id,
      username: req.session.user.username,
      gameId: Number(gameId),
      rating: Number(rating),
      text,

```

```

        status: "published",
        createdAt: new Date().toISOString()
    });
}

writeDB(db);
res.redirect(`/games/${gameId}`);
});

app.post("/complaints/add", requireAuth, (req, res) => {
    const db = readDB();
    const { reviewId, reason } = req.body;

    db.complaints.push({
        id: Date.now(),
        reviewId: Number(reviewId),
        userId: req.session.user.id,
        reason,
        status: "new",
        createdAt: new Date().toISOString()
    });

    writeDB(db);
    res.redirect("/catalog");
});

app.get("/moderator", requireAuth, requireRole("moderator"), (req, res) => {
    const db = readDB();

    const complaints = db.complaints.map((complaint) => {
        const review = db.reviews.find((item) => item.id === complaint.reviewId);

        return {
            ...complaint,
            review
        };
    });

    res.render("moderator", {
        user: req.session.user,
        complaints
    });
});

app.post("/moderator/review/:id/status", requireAuth, requireRole("moderator"),
(req, res) => {
    const db = readDB();
    const review = db.reviews.find((item) => item.id === Number(req.params.id));

    if (review) {
        review.status = req.body.status;
        review.moderatedAt = new Date().toISOString();
    }

    writeDB(db);
    res.redirect("/moderator");
});

app.get("/admin", requireAuth, requireRole("admin"), (req, res) => {
    const db = readDB();

    res.render("admin", {
        user: req.session.user,

```

```

    games: db.games,
    genres: db.genres,
    platforms: db.platforms
  });
});

app.post("/admin/games/add", requireAuth, requireRole("admin"), (req, res) => {
  const db = readDB();

  db.games.push({
    id: Date.now(),
    title: req.body.title,
    year: Number(req.body.year),
    description: req.body.description,
    ageRating: req.body.ageRating,
    developer: req.body.developer,
    publisher: req.body.publisher,
    genres: req.body.genres.split(",").map((item) => item.trim()),
    platforms: req.body.platforms.split(",").map((item) => item.trim())
  });

  writeDB(db);
  res.redirect("/admin");
});

app.get("/analytics", requireAuth, (req, res) => {
  const db = readDB();

  const topGames = db.games
    .map((game) => {
      const rating = getGameRating(game.id, db.reviews);

      return {
        title: game.title,
        rating: rating.average,
        reviewsCount: rating.count
      };
    })
    .sort((a, b) => b.rating - a.rating)
    .slice(0, 10);

  res.render("analytics", {
    user: req.session.user,
    topGames,
    usersCount: db.users.length,
    gamesCount: db.games.length,
    reviewsCount: db.reviews.length,
    collectionsCount: db.collections.length
  });
});

app.listen(PORT, () => {
  console.log(`Сервер запущено: http://localhost:${PORT}`);
});

```

ДОДАТОК Б

Лістинг структури локального сховища даних

```
{
  "users": [
    {
      "id": 1,
      "username": "admin",
      "email": "admin@example.com",
      "passwordHash": "$2a$10$exampleAdminHash",
      "role": "admin",
      "status": "active",
      "createdAt": "2026-04-01T10:00:00.000Z"
    },
    {
      "id": 2,
      "username": "moderator",
      "email": "moderator@example.com",
      "passwordHash": "$2a$10$exampleModeratorHash",
      "role": "moderator",
      "status": "active",
      "createdAt": "2026-04-01T10:05:00.000Z"
    },
    {
      "id": 3,
      "username": "gamer",
      "email": "gamer@example.com",
      "passwordHash": "$2a$10$exampleGamerHash",
      "role": "gamer",
      "status": "active",
      "createdAt": "2026-04-01T10:10:00.000Z"
    }
  ],
  "genres": [
    "Action",
    "Adventure",
    "RPG",
    "Shooter",
    "Strategy",
    "Racing",
    "Simulation",
    "Sports",
    "Indie"
  ],
  "platforms": [
    "PC",
    "PlayStation 5",
    "PlayStation 4",
    "Xbox Series X/S",
    "Xbox One",
    "Nintendo Switch"
  ],
  "games": [
    {
      "id": 101,
      "title": "The Witcher 3: Wild Hunt",
      "year": 2015,
      "description": "Рольова відеогра з відкритим світом, розвиненою сюжетною лінією та системою вибору.",
    }
  ]
}
```

```

    "ageRating": "18+",
    "developer": "CD Projekt Red",
    "publisher": "CD Projekt",
    "genres": ["RPG", "Adventure"],
    "platforms": ["PC", "PlayStation 5", "Xbox Series X/S", "Nintendo
Switch"],
    "cover": "/images/witcher3.jpg"
  },
  {
    "id": 102,
    "title": "God of War",
    "year": 2018,
    "description": "Пригодницька гра з елементами екшену, побудована навколо
сюжетного проходження.",
    "ageRating": "18+",
    "developer": "Santa Monica Studio",
    "publisher": "Sony Interactive Entertainment",
    "genres": ["Action", "Adventure"],
    "platforms": ["PC", "PlayStation 4", "PlayStation 5"],
    "cover": "/images/godofwar.jpg"
  },
  {
    "id": 103,
    "title": "Elden Ring",
    "year": 2022,
    "description": "Рольова гра з відкритим світом, високою складністю та
глибокою бойовою системою.",
    "ageRating": "16+",
    "developer": "FromSoftware",
    "publisher": "Bandai Namco Entertainment",
    "genres": ["RPG", "Action"],
    "platforms": ["PC", "PlayStation 5", "Xbox Series X/S"],
    "cover": "/images/eldenring.jpg"
  }
],
"collections": [
  {
    "id": 501,
    "userId": 3,
    "gameId": 101,
    "copyType": "digital",
    "status": "completed",
    "hoursPlayed": 142,
    "note": "Основна сюжетна лінія завершена, залишилися додаткові завдання.",
    "addedAt": "2026-04-10T14:20:00.000Z"
  },
  {
    "id": 502,
    "userId": 3,
    "gameId": 103,
    "copyType": "physical",
    "status": "playing",
    "hoursPlayed": 38,
    "note": "Фізичне видання для PS5.",
    "addedAt": "2026-04-12T16:45:00.000Z"
  }
],
"wishlist": [
  {
    "id": 701,
    "userId": 3,

```

```

    "gameId": 102,
    "addedAt": "2026-04-13T09:30:00.000Z"
  },
],
"reviews": [
  {
    "id": 801,
    "userId": 3,
    "username": "gamer",
    "gameId": 101,
    "rating": 10,
    "text": "Гра має сильний сюжет, якісний світ і велику кількість
додаткового контенту.",
    "status": "published",
    "createdAt": "2026-04-15T18:10:00.000Z"
  },
  {
    "id": 802,
    "userId": 3,
    "username": "gamer",
    "gameId": 103,
    "rating": 9,
    "text": "Цікава бойова система, складні боси та добре опрацьована
атмосфера.",
    "status": "published",
    "createdAt": "2026-04-16T12:00:00.000Z"
  }
],
"complaints": [
  {
    "id": 901,
    "reviewId": 802,
    "userId": 3,
    "reason": "Перевірити коректність змісту відгуку.",
    "status": "new",
    "createdAt": "2026-04-17T11:15:00.000Z"
  }
]
}

```

ДОДАТОК В

Лістинг клієнтських файлів інтерфейсу

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>GameRate Collection</title>
  <link rel="stylesheet" href="/css/style.css">
</head>
<body>
  <header class="header">
    <div class="container header-row">
      <a href="/" class="logo">GameRate Collection</a>

      <nav class="nav">
        <a href="/catalog">Каталог</a>
        <a href="/analytics">Аналітика</a>
        <a href="/collection">Колекція</a>
        <a href="/login">Вхід</a>
      </nav>
    </div>
  </header>

  <main class="container page">
    <section class="hero">
      <div class="hero-content">
        <h1>Система колекціонування та оцінювання відеоігор</h1>
        <p>
          Програмне забезпечення призначене для ведення особистої ігрової
          колекції, додавання цифрових і фізичних копій, оцінювання відеоігор,
          написання відгуків та перегляду рейтингової інформації.
        </p>

        <div class="hero-actions">
          <a href="/catalog" class="btn primary">Перейти до каталогу</a>
          <a href="/register" class="btn secondary">Створити акаунт</a>
        </div>
      </div>

      <div class="stats-grid">
        <div class="stat-card">
          <span class="stat-value"><%= gamesCount %></span>
          <span class="stat-label">ігор у каталозі</span>
        </div>

        <div class="stat-card">
          <span class="stat-value"><%= reviewsCount %></span>
          <span class="stat-label">відгуків</span>
        </div>

        <div class="stat-card">
          <span class="stat-value"><%= collectionsCount %></span>
          <span class="stat-label">записів колекцій</span>
        </div>

        <div class="stat-card">
          <span class="stat-value"><%= usersCount %></span>
```

```

        <span class="stat-label">користувачів</span>
      </div>
    </div>
  </section>
</main>
</body>
</html>
document.addEventListener("DOMContentLoaded", () => {
  const searchInput = document.querySelector("#searchInput");
  const gameCards = document.querySelectorAll(".game-card");

  if (searchInput) {
    searchInput.addEventListener("input", () => {
      const value = searchInput.value.toLowerCase().trim();

      gameCards.forEach((card) => {
        const title = card.dataset.title.toLowerCase();

        if (title.includes(value)) {
          card.style.display = "block";
        } else {
          card.style.display = "none";
        }
      });
    });
  }

  const ratingInputs = document.querySelectorAll(".rating-input");

  ratingInputs.forEach((input) => {
    input.addEventListener("input", () => {
      const value = Number(input.value);

      if (value < 1) {
        input.value = 1;
      }

      if (value > 10) {
        input.value = 10;
      }
    });
  });

  const copyTypeSelect = document.querySelector("#copyType");
  const digitalFields = document.querySelector(".digital-fields");
  const physicalFields = document.querySelector(".physical-fields");

  if (copyTypeSelect) {
    copyTypeSelect.addEventListener("change", () => {
      if (copyTypeSelect.value === "digital") {
        digitalFields.style.display = "block";
        physicalFields.style.display = "none";
      } else {
        digitalFields.style.display = "none";
        physicalFields.style.display = "block";
      }
    });
  }
});
* {
  box-sizing: border-box;
}

```

```

body {
  margin: 0;
  font-family: Arial, sans-serif;
  background: #0f1117;
  color: #f5f7fb;
}

.container {
  width: min(1180px, 92%);
  margin: 0 auto;
}

.header {
  background: #151923;
  border-bottom: 1px solid #262b36;
}

.header-row {
  display: flex;
  align-items: center;
  justify-content: space-between;
  height: 72px;
}

.logo {
  color: #ffffff;
  text-decoration: none;
  font-size: 22px;
  font-weight: 700;
}

.nav {
  display: flex;
  gap: 22px;
}

.nav a {
  color: #b7c0d4;
  text-decoration: none;
  font-size: 15px;
}

.nav a:hover {
  color: #ffffff;
}

.page {
  padding: 42px 0;
}

.hero {
  display: grid;
  grid-template-columns: 1.2fr 0.8fr;
  gap: 32px;
  align-items: center;
}

.hero-content {
  background: #171c26;
  border: 1px solid #2a3140;
  border-radius: 18px;
  padding: 36px;
}

```

```

.hero-content h1 {
  margin: 0 0 18px;
  font-size: 36px;
  line-height: 1.2;
}

.hero-content p {
  color: #c2cadb;
  font-size: 17px;
  line-height: 1.6;
}

.hero-actions {
  display: flex;
  gap: 14px;
  margin-top: 28px;
}

.btn {
  display: inline-block;
  padding: 12px 18px;
  border-radius: 10px;
  text-decoration: none;
  font-weight: 600;
}

.btn.primary {
  background: #6c63ff;
  color: #ffffff;
}

.btn.secondary {
  background: #252b38;
  color: #ffffff;
  border: 1px solid #343b4b;
}

.stats-grid {
  display: grid;
  grid-template-columns: repeat(2, 1fr);
  gap: 18px;
}

.stat-card {
  background: #171c26;
  border: 1px solid #2a3140;
  border-radius: 16px;
  padding: 26px;
}

.stat-value {
  display: block;
  font-size: 34px;
  font-weight: 700;
  color: #ffffff;
}

.stat-label {
  display: block;
  margin-top: 8px;
  color: #aeb8cc;
}

```

```

.catalog-toolbar {
  display: flex;
  gap: 12px;
  margin-bottom: 24px;
}

.catalog-toolbar input,
.catalog-toolbar select {
  width: 100%;
  padding: 12px;
  background: #171c26;
  color: #ffffff;
  border: 1px solid #2a3140;
  border-radius: 10px;
}

.games-grid {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  gap: 20px;
}

.game-card {
  background: #171c26;
  border: 1px solid #2a3140;
  border-radius: 16px;
  overflow: hidden;
}

.game-card img {
  width: 100%;
  height: 220px;
  object-fit: cover;
}

.game-card-content {
  padding: 16px;
}

.game-title {
  margin: 0 0 8px;
  font-size: 18px;
}

.game-meta {
  color: #aeb8cc;
  font-size: 14px;
}

.rating-badge {
  display: inline-block;
  margin-top: 12px;
  padding: 6px 10px;
  background: #262b36;
  border-radius: 8px;
  color: #ffffff;
}

.form-card {
  max-width: 430px;
  margin: 40px auto;
  background: #171c26;

```

```

    border: 1px solid #2a3140;
    border-radius: 18px;
    padding: 28px;
}

.form-card label {
    display: block;
    margin-bottom: 8px;
    color: #c2cadb;
}

.form-card input,
.form-card textarea,
.form-card select {
    width: 100%;
    padding: 12px;
    margin-bottom: 16px;
    background: #0f1117;
    color: #ffffff;
    border: 1px solid #2a3140;
    border-radius: 10px;
}

.form-card button {
    width: 100%;
    padding: 12px;
    background: #6c63ff;
    color: #ffffff;
    border: none;
    border-radius: 10px;
    font-weight: 700;
    cursor: pointer;
}

.table {
    width: 100%;
    border-collapse: collapse;
    background: #171c26;
    border-radius: 14px;
    overflow: hidden;
}

.table th,
.table td {
    padding: 14px;
    border-bottom: 1px solid #2a3140;
    text-align: left;
}

.table th {
    color: #ffffff;
    background: #1d2330;
}

.table td {
    color: #c2cadb;
}

@media (max-width: 900px) {
    .hero {
        grid-template-columns: 1fr;
    }
}

```

```
.games-grid {
  grid-template-columns: repeat(2, 1fr);
}

.nav {
  gap: 12px;
}
}

@media (max-width: 560px) {
  .header-row {
    height: auto;
    padding: 18px 0;
    flex-direction: column;
    gap: 14px;
  }

  .games-grid,
  .stats-grid {
    grid-template-columns: 1fr;
  }

  .hero-content h1 {
    font-size: 28px;
  }
}
```

ДОДАТОК Г

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Сапега Станіслав Олександрович, здобувач(ка) НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота (проект) на тему «Програмне забезпечення з колекціонування та оцінювання відеоігор за рейтингами користувачів» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):
 - [] інструменти генеративного ШІ не використовувалися.
 - [+] інструменти ШІ (вказати назву: **ChatGPT**, Gemini, Claude, DeepL, _____ інші (вказати назву)) були використані для:
 - [] перекладу іншомовних джерел;
 - [+] стилістичного редагування та перевірки граматики;
 - [] допомоги у написанні/відлагодженні програмного коду;
 - [] інше: __пошуку літератури, допомоги з ідеями_____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«__» _____ 2026 р. _____ Станіслав Сапега
(підпис)