

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КІЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
*Факультет інформатики та обчислювальної техніки***

(повне найменування інституту, факультету)

Інформатики та програмної інженерії

(повна назва кафедри)

**Курсова робота
з дисципліни Компоненти програмної інженерії**

на тему Веб-застосутоок «Ігрова бібліотека»

Виконав (ла): студент (ка) III курсу, групи Ткаченко Костянтин Олександрович III-331

(прізвище, ім'я, по батькові)

(підпис)

ст. викладач, PhD, Смілянець
Ф. А.

(підпис)

посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові

Керівник ст. викладач, PhD, Смілянець

Ф. А.

(підпис)

посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові

Ілени комісії ст. викладач, PhD, Смілянець

Ф. А.

(підпис)

посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові

ст. викладач, PhD, Смілянець
Ф. А.

(підпис)

посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові

Засвідчую, що у цій курсовій роботі
немає запозичень з праць інших
авторів без відповідних посилань.

Студент

(підпис)

Київ – 2026

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення
інформаційних систем

ЗАТВЕРДЖУЮ

керівник

Федір Смілянець

(підпис) (ім'я прізвище)

“ ” 2024 р.

**ЗАВДАННЯ
на курсову роботу студенту**

(прізвище, ім'я, по батькові)

1. Тема роботи Веб-застосунок «Ігрова бібліотека»

керівник роботи Смілянець Федір Андрійович, PhD

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

2. Термін подання студентом роботи « 2 » січня 2025 року

3. Вихідні дані до роботи: технічне завдання

4. Зміст пояснівальної записки

ВСТУП

1. ПЕРЕДПРОЕКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

1.2 Аналіз існуючих рішень

1.2.1 Аналіз відомих програмних продуктів

1.2.2 Аналіз відомих алгоритмічних та технічних рішень

1.3 Аналіз та моделювання бізнес-процесів

ВИСНОВОК ДО РОЗДІЛУ 1

2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Варіанти використання програмного забезпечення

2.2 Розроблення функціональних вимог

2.3 Розроблення нефункціональних вимог

2.4 Аналіз системних вимог

2.5 Постановка завдання на розробку програмного забезпечення

ВИСНОВОК ДО РОЗДІЛУ 2

3. КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

3.2 Архітектурні рішення та обґрунтування вибору засобів розробки

3.3 Конструювання програмного забезпечення

3.3.2 Опис структури бази даних

ВИСНОВОК ДО РОЗДІЛУ 3

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз якості ПЗ 37

4.2 Опис процесів тестування

4.3 Опис контрольного прикладу

ВИСНОВОК ДО РОЗДІЛУ 4

5 ПОВНА ІНСТРУКЦІЯ КОРИСТУВАЧА

5.1 Встановлення програмного забезпечення

5.2 Інструкція користувача

ВИСНОВОК ДО РОЗДІЛУ 5

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

5. Перелік графічного матеріалу

- 1) Діаграма варіантів використання
- 2) BPMN-діаграма бізнес-процесів
- 3) С4-діаграми (Level 1–4)
- 4) ER-діаграма бази даних

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «8» жовтня 2025 року

Календарний план

№ з/п	Назва етапів виконання курсової роботи	Термін виконання етапів роботи	Примітка
1	Вивчення рекомендованої літератури	3.12.2025	
2	Вивчення предметної області	5.12.2025	
3	Аналіз аналогів	7.12.2025	
4	Формування вимог	9.12.2025	
5	Проектування архітектури	12.12.2025	
6	Реалізація backend	15.12.2025	
7	Реалізація frontend	20.12.2025	
8	Тестування	25.12.2025	
9	Оформлення пояснівальної записки	13.01.2025	
10	Подання КР на основний захист	20.02.2025	

Студент Ткаченко Костянтин
(ініціали, прізвище)

Керівник Федір СМІЛЯНЕЦЬ
(ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка курсової роботи складається з п'яти розділів, Загальний обсяг роботи становить близько 53 сторінки.

Курсова робота присвячена розробці веб-застосунку «Ігрова бібліотека» – персональної платформи для відстеження ігрової бібліотеки, статусів ігор, оцінок, коментарів, створення списків та взаємодії з друзями.

Мета роботи – спроектувати, розробити та протестувати повноцінний веб-застосунок на базі фреймворку Django з інтеграцією зовнішнього API RAWG для отримання даних про ігри.

У розділі 1 проведено аналіз предметної області, існуючих рішень (Steam, RAWG, Backloggd, Grouvee) та моделювання бізнес-процесів. Розділ 2 містить постановку функціональних та нефункціональних вимог, варіанти використання та чітке завдання на розробку. У розділі 3 описано архітектуру (MVT), вибір технологій (Python, Django, Tailwind CSS, SQLite), структуру бази даних та повну реалізацію. Розділ 4 присвячено аналізу якості, мануальному тестуванню (10 тест-кейсів) та контролльному прикладу. Розділ 5 надає повну інструкцію користувача та адміністратора з встановлення та використання застосунку.

Програмне забезпечення успішно розгорнуто локально (`runserver`) та протестовано в браузерах Chrome та Firefox.

КЛЮЧОВІ СЛОВА: веб-застосунок, ігрова бібліотека, Django, Python, RAWG API, списки ігор, система друзів, автентифікація, тестування, SQLite, Tailwind CSS.

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

керівник

_____ Максим ГОЛОВЧЕНКО

“ ____ ” _____ 2025 р.

ТЕМА КУРСОВОЇ РОБОТИ

Веб-застосунок «Ігрова бібліотека»

КПІ.ІП-з3116.045440.01.91

“ПОГОДЖЕНО”

Керівник роботи:

_____ Федір СМІЛЯНЕЦЬ

Виконавець:

_____ Ткаченко Костянтин

Київ – 2026

Зміст

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ	8
2. ПІДСТАВА ДЛЯ РОЗРОБКИ.....	9
3. ПРИЗНАЧЕННЯ РОЗРОБКИ.....	10
4. ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	11
4.1 Вимоги до функціональних характеристик	11
4.1 Вимоги до функціональних характеристик	13
4.1.1 Користувачького інтерфейсу	13
4.1.2 Для неавторизованого користувача	14
4.1.3 Для авторизованого користувача	14
4.2 Вимоги до надійності	14
4.3 Умови експлуатації	14
4.4 Вимоги до технічних засобів	15
4.5 Вимоги до інформаційної та програмної сумісності	15
5. ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ	16
5.1 Попередній склад програмної документації	16
5.2 Спеціальні вимоги до програмної документації	16
6 СТАДІЇ І ЕТАПИ РОЗРОБКИ	17
7. ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ	18

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Веб-застосування «Ігрова бібліотека».

Галузь застосування: Наведене технічне завдання поширюється на розробку програмного забезпечення «Ігрова бібліотека», котра використовується для організації та відстеження особистих колекцій відеоігор, надання інструментів для оцінки, рекомендацій та соціальної взаємодії між користувачами в сфері геймінгу та призначена для геймерів-ентузіастів, які бажають ефективно керувати своїм ігровим контентом, планувати проходження ігор та ділитися досвідом у спільноті.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки веб-застосунку «Ігрова бібліотека» є індивідуальний

навчальний план студента та РНП затверджена кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для автоматизованого ведення та управління персональною колекцією відеоігор, відстеження прогресу проходження, оцінки ігор, написання відгуків, зручного пошуку ігор через RAWG API та соціальної взаємодії між користувачами.

Метою розробки є створення зручного, швидкого та персоналізованого веб-застосунку, який полегшить геймерам організацію ігрового контенту, підвищить залученість до хобі завдяки соціальним функціям та забезпечить якісні рекомендації нових ігор на основі статистики користувача. Основні показники якості: швидкість роботи інтерфейсу, точність рекомендацій, повнота функціоналу та рівень задоволеності користувачів.

4. ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функцій:

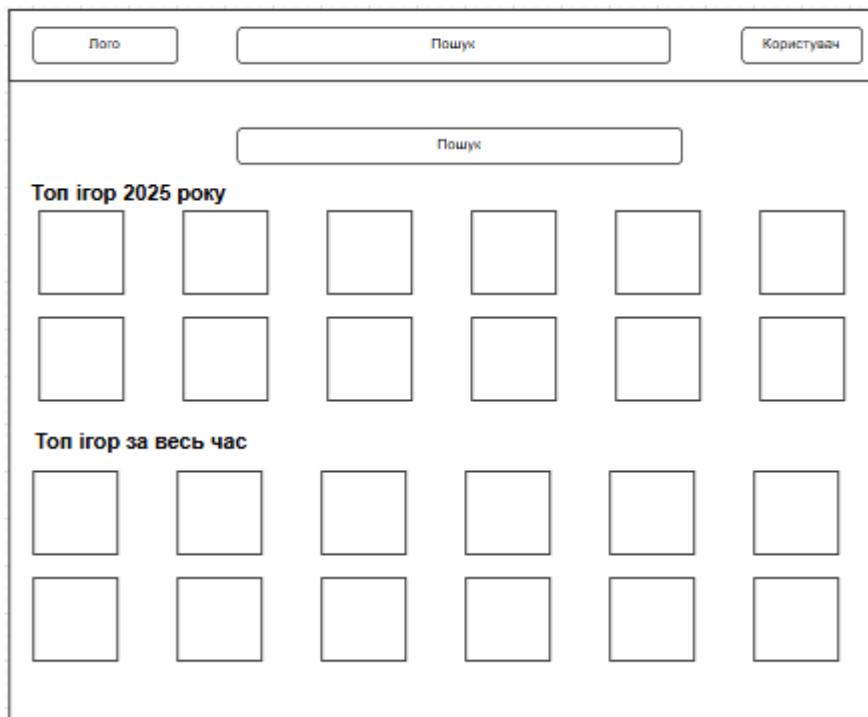


Рис 4.1 Прототип головної сторінки

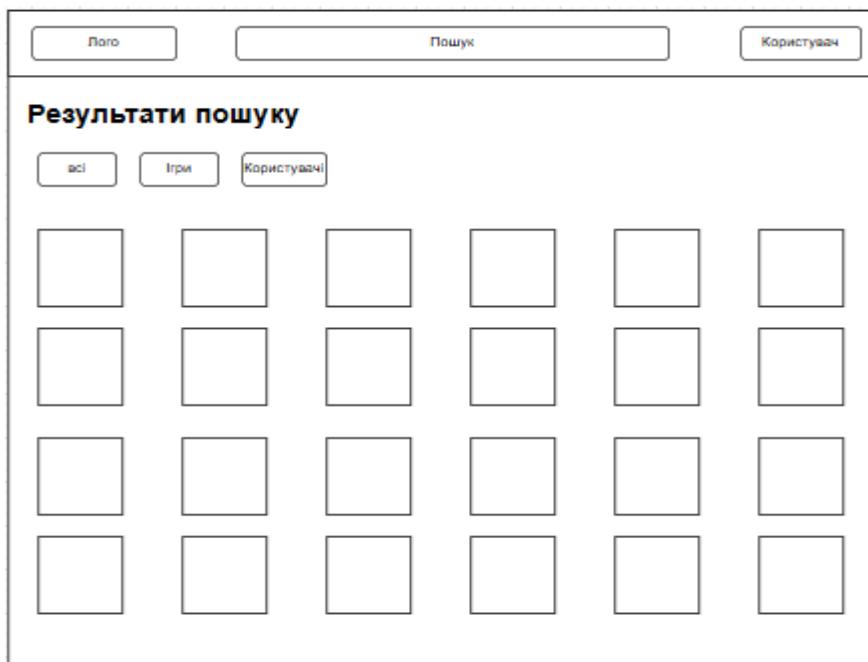


Рис 4.2 Прототип сторінки пошуку

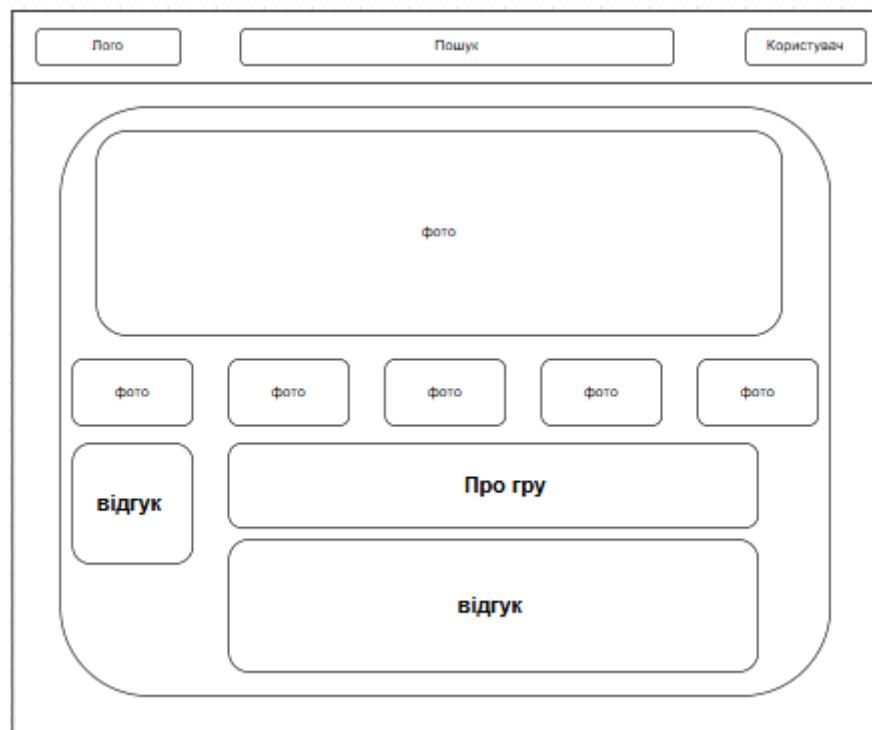


Рис 4.3 Прототип сторінки гри

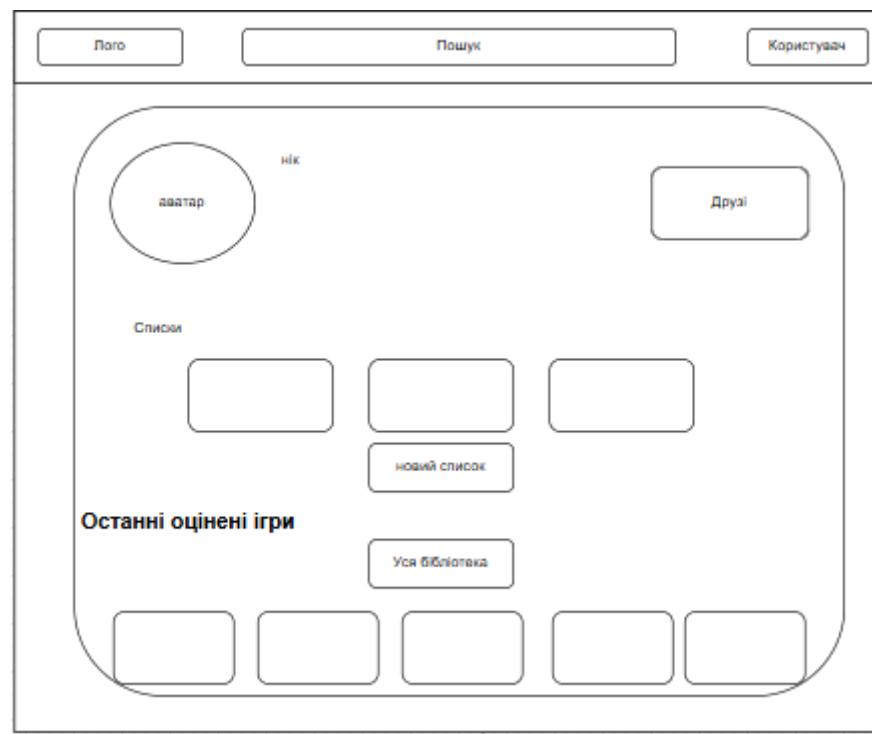


Рис 4.4 Прототип сторінки користувача

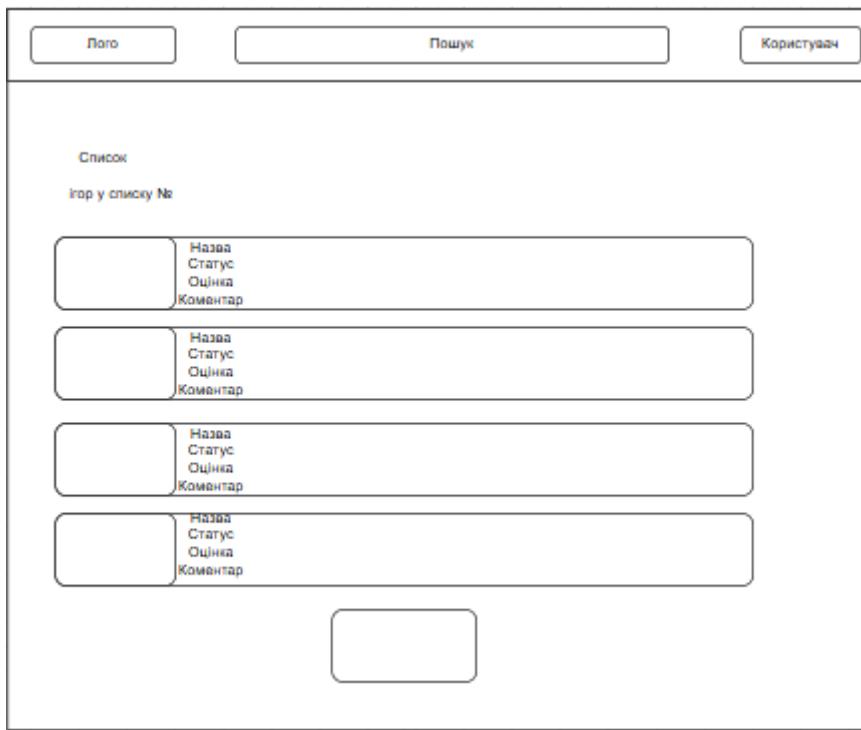


Рис 4.5 Прототип сторінки списку

4.1 Вимоги до функціональних характеристик

4.1.1 Користувальського інтерфейсу

Інтерфейс має бути повністю респонсивним. Підтримуються світла та темна теми. Усі основні дії (пошук, додавання гри, зміна статусу, оцінка) виконуються не більше ніж за 3 кліки. Використовуються сучасні UI-елементи: картки ігор з обкладинками, модальні вікна, градієнти, hover-ефекти, backdrop-blur, анімації, інтерактивна галерея медіа.

Основні екрани (прототипи UX):

- головна сторінка (рекомендації, новинки, топ, очікувані релізи);
- детальна сторінка гри (опис, галерея, характеристики, оцінки, відгуки);
- сторінка пошуку (вкладки: Всі / Ігри / Користувачі);
- профіль користувача (аватар, статистика, списки, друзі, запити);
- авторизація / реєстрація;
- сторінка списку ігор;
- форми додавання/редагування статусу, оцінки, відгуку та списку.

4.1.2 Для неавторизованого користувача

Перегляд головної сторінки, секцій ігор, пошуку, детальної інформації про гру, публічних профілів та публічних списків інших користувачів, перехід на сторінку реєстрації/авторизації.

4.1.3 Для авторизованого користувача

Усі можливості неавторизованого користувача +:

- виставлення оцінки грі (1–5 зірок);
- зміна статусу проходження («Граю», «Пройдено», «У планах», «Покинуто»);
- написання, редагування, видалення власного відгуку;
- додавання гри до бібліотеки та власних списків;
- створення, редагування, видалення власних списків (публічні/приватні);
- завантаження та зміна аватара;
- надсилання/прийняття/відхилення запитів у друзі, видалення з друзів;
- перегляд відгуків та статусів друзів на сторінках ігор;
- перегляд бібліотеки та списків друзів (якщо публічні).

4.2 Вимоги до надійності

Передбачено валідацію введених даних та захист від некоректних дій.

Забезпечується цілісність даних у базі (автозбереження при втраті з'єднання). Час безперебійної роботи сервера — не менше 99 % (в рамках локального розгортання). Помилки логуються на сервері.

4.3 Умови експлуатації

Запуск та експлуатація — на локальному сервері розробника.

Обслуговування здійснює розробник (періодичні оновлення, усунення збоїв). Постійний адміністратор не потрібен.

4.4 Вимоги до технічних засобів

ІБМ-сумісний ПК. Мінімум: Intel Core i3 / аналог, 4 ГБ RAM, інтернет від 20 Мбіт/с. Рекомендовано: Ryzen 5 / Core i5 або вище, 8 ГБ RAM, інтернет від 100 Мбіт/с.

4.5 Вимоги до інформаційної та програмної сумісності

Вхідні дані: JSON від RAWG API, дані форм (текст, числа, файли аватарів .jpg/.png/.webp/.gif), рядки пошуку. Усі дані валіduються, екрануються від XSS та SQL-ін'єкцій.

Вихідні дані: HTML5-сторінки (Tailwind CSS + JS), JSON для API, зображення (.webp/.jpg/.png), UTF-8 текст з базовим markdown.

Мова розробки:

- Backend — Python 3.10+ + Django 4.x/5.x
- Frontend — HTML5, CSS3 (Tailwind), JavaScript ES6+

Середовище: PyCharm, Git + GitHub, npm/pnpm, Django runserver, SQLite (розробка), браузери Chrome/Zen.

Вихідний код: структурований Django-проект (apps: accounts, games, lists, friends тощо), коментарі до складних частин, компонентні шаблони.

Спеціальні вимоги Не висуваються.

5. ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

- технічне завдання;
- пояснівальна записка;
- текст програми.

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

7. ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Розділу тестування програмного забезпечення”.

Пояснювальна записка до курсової роботи
на тему: Онлайн ігрова бібліотека
КП.ІП-з3116.045440.02.81

Київ – 2025

Зміст

ВСТУП	22
1. ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ.....	23
1.1 Аналіз предметної області	23
1.2 Аналіз існуючих рішень	24
1.2.1 Аналіз відомих програмних продуктів	24
1.2.2 Аналіз відомих алгоритмічних та технічних рішень	25
1.3 Аналіз та моделювання бізнес-процесів.....	25
ВИСНОВОК ДО РОЗДІЛУ 1.....	27
2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	28
2.1 Варіанти використання програмного забезпечення	28
2.2 Розроблення функціональних вимог	36
2.3 Розроблення нефункціональних вимог	41
2.4 Аналіз системних вимог	42
2.5 Постановка завдання на розробку програмного забезпечення	42
ВИСНОВОК ДО РОЗДІЛУ 2.....	43
3. КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	44
3.1 Архітектура програмного забезпечення	44
3.2 Архітектурні рішення та обґрунтування вибору засобів розробки	47
3.3 Конструювання програмного забезпечення.....	48
3.3.2 Опис структури бази даних.....	49
ВИСНОВОК ДО РОЗДІЛУ 3.....	53
4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	55
4.1 Аналіз якості ПЗ.....	55
4.2 Опис процесів тестування	56
4.3 Опис контрольного прикладу.....	59
ВИСНОВОК ДО РОЗДІЛУ 4.....	61
5 ПОВНА ІНСТРУКЦІЯ КОРИСТУВАЧА	62
5.1 Встановлення програмного забезпечення	62
5.2 Інструкція користувача	63
ВИСНОВОК ДО РОЗДІЛУ 5.....	66
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	68

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface – прикладний програмний інтерфейс.

БД – База даних.

ER – Entity-Relationship diagram – діаграма сутностей-зв'язків.

IDE – Integrated Development Environment – інтегроване середовище розробки.

ІТ – Інформаційні технології.

OS – Операційна система (англ. Operating System).

SDK – Software Development Kit – набір інструментів для розробки програмного забезпечення.

RAWG – База даних ігор (API для отримання даних про ігри).

Django – Фреймворк для веб-розробки на Python.

HTTP – HyperText Transfer Protocol – протокол передачі гіпертексту.

URL – Uniform Resource Locator – уніфікований локатор ресурсів.

JSON – JavaScript Object Notation – формат обміну даними.

REST – Representational State Transfer – архітектурний стиль для API

ВСТУП

Розвиток ігрової індустрії та зростання кількості відеоігор створюють потребу в зручних цифрових інструментах для каталогізації, оцінки та соціальної взаємодії навколо ігрових бібліотек. Існуючі платформи часто обмежені комерційними інтересами, недостатньою приватністю списків або слабкою соціальною складовою для україномовних користувачів.

Актуальність теми зумовлена необхідністю створення персоналізованого веб-застосунку, який інтегрує відкритий API RAWG, підтримує статуси ігор, оцінки, коментарі, приватні/публічні списки та систему друзів з відгуками.

Мета роботи — розробити веб-застосунок «Ігрова бібліотека» на базі Django з інтеграцією RAWG API для управління персональною ігровою колекцією та соціальною взаємодією.

Завдання:

- проаналізувати предметну область та аналоги;
- спроектувати моделі даних та архітектуру;
- реалізувати backend (моделі, views, API-інтеграцію);
- розробити інтерфейс та функціонал автентифікації, профілів, списків, статусів, друзів;
- провести тестування основних сценаріїв.

Об'єкт дослідження — процеси управління ігровою колекцією та соціальної взаємодії геймерів у веб-середовищі.

Предмет дослідження — архітектура та реалізація веб-застосунку «Ігрова бібліотека».

Методи — аналіз, моделювання, об'єктно-орієнтоване програмування, веб-розробка (Django, REST API).

1. ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Предметною областю курсової роботи є автоматизація ведення персональної ігрової бібліотеки та соціальної взаємодії геймерів у веб-середовищі. Основними процесами є:

- пошук та отримання інформації про відеоігри;
- відстеження статусів проходження ігор (граю, пройдено, у планах, відкладено);
- оцінка, коментування та організація ігор у списки;
- обмін досвідом із друзями (перегляд відгуків друзів, система запитів у друзі);
- формування персональних профілів та публічних/приватних колекцій.

На сьогодні більшість гравців використовують або вбудовані функції платформ (Steam, Epic Games, PlayStation Network), або сторонні сервіси (RAWG, Backloggd, HowLongToBeat), які часто мають обмеження: комерційну спрямованість, відсутність глибокої приватності списків, слабку соціальну складову для невеликих спільнот або недостатню україномовну підтримку.

Розробка веб-застосунку «Ігрова бібліотека» дозволяє створити зручний, персоналізований та соціально-орієнтований інструмент, інтегрований з відкритим API RAWG, з акцентом на приватність, гнучкість списків та відображення активності друзів.

1.2 Аналіз існуючих рішень

1.2.1 Аналіз відомих програмних продуктів

Для порівняння функціональності розроблюваного застосунку з аналогами використано таблицю 1.3.

Таблиця 1.3 – Порівняння з аналогами

№	Функція	«Ігрова бібліотека»	Backlogd	Steam
1	Реєстрація та автентифікація	+	+	+
2	Пошук ігор з фільтрами	+	+	+
3	Детальна сторінка гри (опис, скріншоти, трейлери з API)	+	+	+
4	Статуси ігор (граю / пройдено / у планах / відкладено)	+	+	-
5	Оцінка ігор (1–5)	+	+	+
6	Коментарі / відгуки до ігор	+	+	+
7	Приватні та публічні списки ігор	+	+	+
8	Система друзів + надсилання / прийняття запитів	+	-	+
9	Перегляд профілів інших користувачів (з урахуванням приватності)	+	+	+

10	Відгуки друзів безпосередньо на сторінці гри	+	+	+
11	Головна сторінка з топами, новинками, очікуваними релізами	+	-	+
12	Завантаження та зміна аватара профілю	+	+	+

Аналіз показує, що розроблюваний застосунок поєднує переваги спеціалізованих трекерів (Backloggd) та соціальних платформ (Steam), додаючи при цьому сильну приватність списків, відгуки друзів на сторінці гри та повну україномовну підтримку.

1.2.2 Аналіз відомих алгоритмічних та технічних рішень

Для реалізації застосунку обрано клієнт-серверну архітектуру з використанням фреймворку Django (Python). Зберігання даних здійснюється в реляційній базі даних PostgreSQL (або SQLite для розробки). Обмін даними з зовнішнім сервісом RAWG реалізовано через REST API. Інтерфейс користувача буде створений за допомогою шаблонів Django, HTML5, CSS (з можливим підключенням Bootstrap/Tailwind) та JavaScript. Автентифікація — вбудована система Django Authentication. Соціальні функції (друзі, запити) реалізовані через моделі Profile, FriendRequest та зв'язки ManyToMany.

1.3 Аналіз та моделювання бізнес-процесів

Ключовими бізнес-процесами, які необхідно автоматизувати в рамках веб-застосунку «Ігрова бібліотека», є:

- пошук та перегляд інформації про гру;
- додавання гри до персональної бібліотеки;

- встановлення статусу проходження, оцінки та текстового відгуку;
- збереження змін на сервері з валідацією даних;
- відображення оновленої інформації на сторінці гри.

Найважливішим і найчастіше використовуваним сценарієм є процес «Записати відгук про гру» (встановлення статусу, оцінки та коментаря). Для його формалізації використано нотацію BPMN.

На рисунку 1.1 представлено BPMN-діаграму основного бізнес-процесу «Записати відгук про гру».

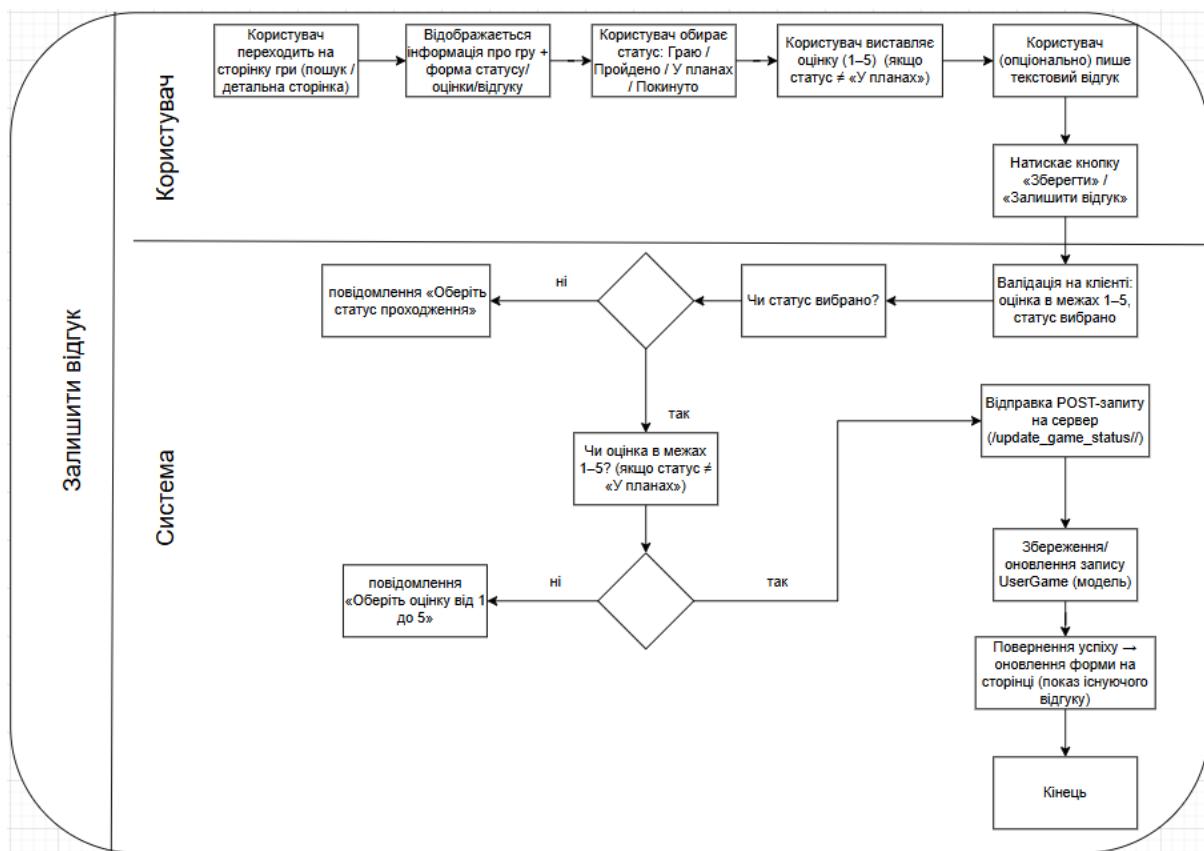


Рисунок 1.1 – Схема бізнес-процесу «Записати відгук про гру»

Опис послідовності дій:

1. Користувач переходить на сторінку гри (через пошук або посилання).
2. Система відображає детальну інформацію про гру та форму для введення статусу, оцінки та відгуку.

3. Користувач обирає статус проходження (Граю / Пройдено / У планах / Відкладено).
4. Якщо статус \neq «У планах», користувач має можливість виставити оцінку від 1 до 10.
5. Користувач (опціонально) вводить текстовий коментар.
6. Користувач натискає кнопку «Зберегти».
7. На клієнтській стороні виконується валідація:
 - чи обрано статус;
 - чи оцінка (якщо потрібна) в межах 1–10.
8. Якщо валідація не пройдена \rightarrow відображається повідомлення про помилку.
9. Якщо валідація успішна \rightarrow відправляється POST-запит на сервер (view update_game_status).
10. Сервер зберігає або оновлює запис у моделі UserGame.
11. Система повертає користувача на сторінку гри з оновленими даними та повідомленням про успіх.

ВИСНОВОК ДО РОЗДІЛУ 1

Проведено аналіз предметної області — ведення персональної ігрової бібліотеки та соціальної взаємодії геймерів. Визначено основні процеси: пошук ігор, управління статусами та оцінками, створення списків, соціальні функції (друзі, запити, відгуки друзів).

Виконано порівняльний аналіз з аналогами (Backloggd, Steam), що показав конкурентні переваги розробленого застосунку в приватності списків, відгуках друзів на сторінці гри та повній україномовній локалізації. Обрано ключовий бізнес-процес «Записати відгук про гру» та змодельовано його за допомогою BPMN-нотації (рис. 1.1), що дозволяє чітко описати взаємодію користувача з системою, клієнтську та серверну валідацію, а також обробку даних у базі.

2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Варіанти використання програмного забезпечення

Для виявлення та формалізації функціональних вимог до веб-застосунку «Ігрова бібліотека» використано підхід, заснований на моделюванні варіантів використання (Use Case Diagram) за нотацією UML.

На рисунку 2.1 представлено діаграму варіантів використання, яка відображає основних акторів (Неавторизований користувач, Авторизований користувач) та ключові сценарії взаємодії з системою.



Рисунок 2.1 – Діаграма варіантів використання

Основні актори та їх взаємодія з системою:

- **Неавторизований користувач:**
 - Перегляд головної сторінки
 - Пошук ігор
 - Перегляд детальної сторінки гри
 - Перегляд публічних профілів та списків інших користувачів
- **Авторизований користувач:**
 - Реєстрація (для нових користувачів)
 - Авторизація
 - Управління друзями (додавання, видалення, надсилання/прийняття запитів)
 - Написання відгуку (статус, оцінка, коментар) про гру
 - Редагування власного відгуку
 - Перегляд відгуків друзів на сторінці гри
 - Створення списків ігор
 - Додавання гри до списку
 - Видалення гри зі списку
 - Створення / редагування / перейменування списків
 - Зміна аватара профілю

Use case name	Реєстрація нового користувача
Use case ID	UC-01
Goals	Створити новий обліковий запис у системі
Actors	Гість (неавторизований користувач)
Trigger	Користувач бажає зареєструватися
Pre-conditions	-
Flow of Events	1. Користувач переходить на сторінку /register/. 2. Заповнює поля: username, password, password

	confirmation. 3. Натискає «Зареєструватися». 4. Система перевіряє валідність (унікальність username, співпадіння паролів, мінімальна довжина). 5. При успіху створюється User + Profile, автоматичний вхід. 6. Перенаправлення на головну сторінку з повідомленням про успіх.
Extension	Некоректні дані → підсвічування помилок під полями, кнопка неактивна або показує помилку.
Post-conditions	Новий користувач створений та авторизований.

Use case name	Авторизація (вхід)
Use case ID	UC-02
Goals	Отримати доступ до персоналізованих функцій
Actors	Гість
Trigger	Користувач бажає увійти
Pre-conditions	Користувач зареєстрований
Flow of Events	1. Перехід на /login/. 2. Введення username та password. 3. Натиснення «Увійти». 4. Перевірка через Django auth. 5. Успіх → авторизація, перенаправлення на попередню/головну сторінку.
Extension	Невірні дані → повідомлення «Невірний логін або пароль».
Post-conditions	Користувач авторизований.

Use case name	Перегляд головної сторінки
Use case ID	UC-03
Goals	Ознайомитися з топами, новинками та очікуваними релізами

Actors	Неавторизований / Авторизований користувач
Trigger	Відкриття сайту або натискання логотипу/«Головна»
Pre-conditions	-
Flow of Events	1. Система завантажує головну сторінку. 2. Відображаються секції: Топ 2025, Топ за весь час, Гарячі новинки, Очікувані 2026 (з RAWG API). 3. Користувач переглядає списки ігор з обкладинками та назвами.
Extension	-
Post-conditions	Користувач ознайомлений з актуальним контентом.

Use case name	Пошук ігор
Use case ID	UC-04
Goals	Знайти ігри за назвою, жанрами, роками
Actors	Неавторизований / Авторизований користувач
Trigger	Введення запиту в поле пошуку
Pre-conditions	-
Flow of Events	1. Користувач вводить запит та/або фільтри (жанри, роки). 2. Натискає «Пошук» або Enter. 3. Система відправляє запит до RAWG API. 4. Відображається сторінка результатів з пагінацією.
Extension	Порожній результат → повідомлення «Нічого не знайдено».
Post-conditions	Користувач бачить список релевантних ігор.

Use case name	Перегляд детальної сторінки гри
Use case ID	UC-05
Goals	Ознайомитися з повною інформацією про гру
Actors	Неавторизований / Авторизований користувач

Trigger	Клік по грі в пошуку / головній / списку
Pre-conditions	Гра існує в RAWG
Flow of Events	1. Завантаження даних з RAWG API (опис, скріншоти, трейлери, платформи тощо). 2. Відображення інформації. 3. Якщо авторизований → показ форми статусу/оцінки/коментаря + відгуки друзів.
Extension	-
Post-conditions	Користувач отримав детальну інформацію.

Use case name	Перегляд публічних профілів та списків
Use case ID	UC-06
Goals	Ознайомитися з іграми та списками іншого користувача
Actors	Неавторизований / Авторизований користувач
Trigger	Перехід за посиланням на профіль /username/
Pre-conditions	Профіль або список публічний
Flow of Events	1. Завантаження профілю. 2. Відображення останніх ігор, списків (якщо не приватні), друзів (якщо доступно). 3. Можливість переходу до детальних списків.
Extension	Приватний контент → заборона доступу.
Post-conditions	Користувач переглянув публічний контент профілю.

Use case name	Перегляд відгуків друзів на сторінці гри
Use case ID	UC-07
Goals	Побачити оцінки та коментарі друзів до конкретної гри
Actors	Авторизований користувач
Trigger	Перегляд детальної сторінки гри
Pre-conditions	Користувач має друзів, друзі оцінили цю гру

Flow of Events	1. На сторінці гри завантажуються UserGame друзів. 2. Відображаються останні 10 не-порожніх відгуків (статус, оцінка, коментар). 3. Клік по імені друга → перехід до його профілю.
Extension	Немає відгуків → блок не відображається.
Post-conditions	Користувач ознайомлений з думками друзів.

Use case name	Управління друзями
Use case ID	UC-08
Goals	Надсилати запити, приймати/відхиляти, видаляти друзів
Actors	Авторизований користувач
Trigger	Перегляд профілю іншого користувача
Pre-conditions	Авторизований
Flow of Events	1. На профілі → кнопки «Додати в друзі» / «Надіслати запит». 2. Надсилання → створення FriendRequest (pending). 3. На своєму профілі → прийняти/відхилити вхідні запити. 4. Видалення друга → видалення зв'язку та запитів.
Extension	Повторний запит після відхилення.
Post-conditions	Список друзів / запити оновлено.

Use case name	Написання відгуку
Use case ID	UC-09
Goals	Встановити статус, оцінку та коментар до гри
Actors	Авторизований користувач
Trigger	Сторінка гри + бажання записати стан
Pre-conditions	Гра існує

Flow of Events	1. Обрати статус. 2. (Якщо не «У планах») обрати оцінку 1–10. 3. Написати коментар (опціонально). 4. Натиснути «Зберегти». 5. Валідація → збереження в UserGame.
Extension	Некоректна оцінка → помилка.
Post-conditions	Відгук збережений та відображається.

Use case name	Редагування відгуку
Use case ID	UC-10
Goals	Змінити статус, оцінку або коментар
Actors	Авторизований користувач
Trigger	Сторінка гри + існуючий відгук
Pre-conditions	Відгук вже існує
Flow of Events	1. Форма заповнена поточними даними. 2. Зміна полів. 3. «Зберегти» → оновлення UserGame.
Extension	-
Post-conditions	Відгук оновлено.

Use case name	Створення / редагування списків
Use case ID	UC-11
Goals	Створити новий список або змінити існуючий
Actors	Авторизований користувач
Trigger	Кнопка «Створити список» або редагування в профілі
Pre-conditions	Авторизований
Flow of Events	1. Введення назви. 2. Вибір приватності. 3. (Редагування) зміна назви / приватності. 4. Збереження → створення/оновлення UserList.
Extension	Порожня назва → помилка.

Post-conditions	Список створений/оновлений.
-----------------	-----------------------------

Use case name	Додавання гри до списків
Use case ID	UC-12
Goals	Додати гру до одного або кількох списків
Actors	Авторизований користувач
Trigger	Сторінка гри
Pre-conditions	Є хоча б один список
Flow of Events	1. Вибір списку(-ів) або створення нового. 2. «Додати» → зв'язок UserGame ↔ UserList.
Extension	-
Post-conditions	Гра в списку.

Use case name	Видалення гри зі списку
Use case ID	UC-13
Goals	Прибрати гру зі списку
Actors	Авторизований користувач (власник списку)
Trigger	Сторінка списку
Pre-conditions	Гра є в списку
Flow of Events	1. Клік «Видалити» біля гри. 2. Підтвердження (опціонально). 3. Видалення зв'язку UserGame ↔ UserList.
Extension	-
Post-conditions	Гра більше не в списку.

Use case name	Зміна аватарки профілю
Use case ID	UC-14
Goals	Оновити фото профілю

Actors	Авторизований користувач
Trigger	Сторінка профілю → кнопка завантаження
Pre-conditions	Авторизований
Flow of Events	1. Вибір файлу зображення. 2. Натиснення «Завантажити». 3. Збереження в Profile.avatar. 4. Оновлення сторінки.
Extension	Непідтримуваний формат / великий розмір → помилка.
Post-conditions	Аватар оновлено.

2.2 Розроблення функціональних вимог

Програмне забезпечення «Ігрова бібліотека» структуровано за модулями, кожен з яких відповідає за певну групу функцій. Загальна модель функціональних вимог наведена в таблиці 2.15. Детальний опис кожної вимоги подано в таблицях 2.16–2.28. Матрицю трасування вимог (зв’язок між Use Case та функціональними вимогами) подано на рисунку 2.3

Таблиця 2.15 – Загальна модель вимог

№	Назва вимоги	ID	Пріоритет	Ризик
1	Перегляд головної сторінки з топами та новинками	FR-01	Середній	Низький
2	Пошук ігор за назвою, жанрами, роками	FR-02	Високий	Середній

3	Перегляд детальної сторінки гри (інтеграція RAWG API)	FR-03	Високий	Середній
4	Перегляд публічних профілів та списків інших користувачів	FR-04	Середній	Низький
5	Реєстрація нового користувача	FR-05	Високий	Високий
6	Авторизація користувача	FR-06	Високий	Високий
7	Вихід з облікового запису	FR-07	Середній	Низький
8	Написання відгуку про гру (статус, оцінка, коментар)	FR-08	Високий	Середній
9	Редагування власного відгуку	FR-09	Високий	Низький
10	Перегляд відгуків друзів на сторінці гри	FR-10	Високий	Середній
11	Управління друзями (запити, прийняття, видалення)	FR-11	Високий	Середній
12	Зміна аватара профілю	FR-12	Середній	Низький
13	Створення списку ігор	FR-13	Високий	Низький
14	Редагування / перейменування списку	FR-14	Середній	Низький
15	Додавання гри до списку	FR-15	Високий	Низький
16	Видалення гри зі списку	FR-16	Середній	Низький

Таблиця 2.16 – Перелік функціональних вимог

№	ID	Назва вимоги	Опис
1	FR-01	Перегляд головної сторінки	Неавторизований та авторизований користувач бачить головну сторінку з секціями топів ігор 2025 року, топів за весь час, гарячих новинок та очікуваних релізів 2026 (дані з RAWG API).
2	FR-02	Пошук ігор	Користувач може шукати ігри за назвою, жанрами, роками випуску з фільтрами та пагінацією (інтеграція з RAWG API).
3	FR-03	Перегляд детальної сторінки гри	Відображення повної інформації про гру: опис, розробник, платформи, жанри, рейтинг, скріншоти, трейлери (з RAWG API). Для авторизованих — форма статусу, оцінки, коментаря та блок відгуків друзів.
4	FR-04	Перегляд публічних профілів та списків інших користувачів	Доступ до публічних даних профілю: останні ігри, списки (неприватні), друзі (якщо доступно).
5	FR-05	Реєстрація нового користувача	Створення облікового запису з username та паролем, автоматична авторизація після успішної реєстрації.
6	FR-06	Авторизація користувача	Вхід у систему за username та паролем.

7	FR-07	Вихід з облікового запису	Завершення сесії та перенаправлення на головну сторінку.
8	FR-08	Написання відгуку про гру	Встановлення статусу (Граю / Пройдено / У планах / Відкладено), оцінки 1–10 та текстового коментаря.
9	FR-09	Редагування власного відгуку	Зміна статусу, оцінки або коментаря до раніше доданої гри.
10	FR-10	Перегляд відгуків друзів на сторінці гри	Відображення останніх 10 непорожніх відгуків (статус, оцінка, коментар) друзів на сторінці конкретної гри.
11	FR-11	Управління друзями	Надсилання запитів у друзі, прийняття/відхилення вхідних запитів, видалення з друзів.
12	FR-12	Зміна аватара профілю	Завантаження та оновлення фото профілю користувача.
13	FR-13	Створення списку ігор	Створення нового списку з назвою та вибором приватності (приватний/публічний).
14	FR-14	Редагування / перейменування списку	Зміна назви списку або статусу приватності.
15	FR-15	Додавання гри до списку	Додавання гри до одного або кількох існуючих списків (або створення нового під час додавання).
16	FR-16	Видалення гри зі списку	Видалення гри з конкретного списку (без видалення самої гри з бібліотеки).

Рисунок 2.3 – Матриця трасування вимог

2.3 Розроблення нефункціональних вимог

Нефункціональні вимоги визначають якісні характеристики системи, які впливають на її зручність, надійність, продуктивність та безпеку. Вони класифіковані за основними групами:

1. Вимоги до інтерфейсу користувача

- Інтерфейс україномовний, інтуїтивно зрозумілий.
 - Час завантаження сторінки не більше 3 секунд за умови стабільного інтернет-з'єднання.

2. Вимоги до продуктивності

- Час відповіді сервера на запит (без урахування зовнішнього API) — не більше 500 мс.
 - Час завантаження даних з RAWG API — не більше 4 секунд
 - Система повинна витримувати одночасну роботу до 100 активних користувачів без значного падіння продуктивності

3. Вимоги до надійності та доступності

- Доступність системи 24/7 за винятком планового технічного обслуговування.
 - Автоматичне відновлення сесії після помилок мережі
 - Захист від втрати даних: збереження всіх змін статусів, оцінок, коментарів та списків у базі даних.

4. Вимоги до безпеки

- Захист паролів за допомогою хешування
 - Обмеження доступу до приватних списків та профілів

5. Вимоги до масштабованості та розширюваності

- Архітектура дозволяє легко додавати нові функції (наприклад, рекомендації, досягнення, мультиплатформість).
- Використання модульної структури Django

6. Вимоги до сумісності

- Підтримка сучасних браузерів: Chrome, Firefox, Edge, Safari

2.4 Аналіз системних вимог

Системні вимоги до середовища розробки та розгортання:

- Серверна частина: Python 3.10+, Django 4.2+, SQLite
- Зовнішні сервіси: RAWG API
- Мінімальні вимоги до хостингу: 1 CPU, 1 GB RAM, 10 GB диск (для курсової достатньо безкоштовного хостингу типу Render або Railway).
- Розробка ведеться на ОС Windows 11

2.5 Постановка завдання на розробку програмного забезпечення

Метою розробки є створення веб-застосунку «Ігрова бібліотека» — персонального цифрового інструменту для геймерів, який дозволяє вести облік пройдених ігор, планувати майбутні релізи, оцінювати ігри, створювати приватні та публічні списки, а також обмінюватися враженнями з друзями.

Основні цілі проекту:

- Забезпечити зручне україномовне середовище для каталогізації ігрової колекції.
- Інтегрувати відкритий API RAWG для автоматичного отримання актуальних даних про ігри.
- Реалізувати соціальні функції: друзі, запити в друзі, перегляд відгуків друзів безпосередньо на сторінці гри.
- Надати гнучкий контроль приватності

Завдання, що підлягають вирішенню:

- Розробити повноцінний backend на Django з моделями користувачів, ігор, списків, статусів, друзів та запитів.
- Забезпечити інтеграцію з RAWG API для пошуку, детальної інформації, топів та новинок.
- Реалізувати автентифікацію, профілі, списки, відгуки, систему друзів.
- Створити адаптивний frontend-інтерфейс з використанням шаблонів Django та CSS.
- Провести функціональне тестування основних сценаріїв.
- Підготувати документацію та захистити роботу.

ВИСНОВОК ДО РОЗДІЛУ 2

У другому розділі розроблено повний комплект вимог до програмного забезпечення «Ігрова бібліотека».

Визначено функціональні вимоги (таблиця 2.16), які охоплюють усі ключові сценарії: від перегляду головної сторінки та пошуку ігор до управління списками, відгуками та друзьями. Побудовано матрицю трасування (рис. 2.3), яка підтверджує повне покриття вимог варіантами використання.

Сформульовано нефункціональні вимоги за класифікацією: інтерфейс, продуктивність, безпека, надійність, масштабування. Проаналізовано системні вимоги до середовища розробки та розгортання.

Узагальнено постановку завдання: мета — створення персоналізованого україномовного інструменту для ведення ігрової бібліотеки з соціальними функціями та інтеграцією RAWG API.

3. КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

Для веб-застосунку «Ігрова бібліотека» обрано класичну клієнт-серверну архітектуру з використанням архітектурного патерну **Model-View-Template (MVT)**, який є стандартним для фреймворку Django.

Це трирівнева архітектура:

- **Рівень представлення (Presentation Layer)** — шаблони HTML + CSS + JavaScript (фронтенд, що рендериться на сервері).
- **Рівень бізнес-логіки (Application Layer)** — Django views, forms, middleware, сервісні функції, інтеграція з RAWG API.
- **Рівень даних (Data Layer)** — реляційна база даних (PostgreSQL / SQLite), моделі Django ORM.

Додатково застосовується **REST-подібний підхід** для взаємодії з зовнішнім RAWG API (HTTP-запити).

Опис компонентів високорівневої архітектури (рівень 1 C4 Model)

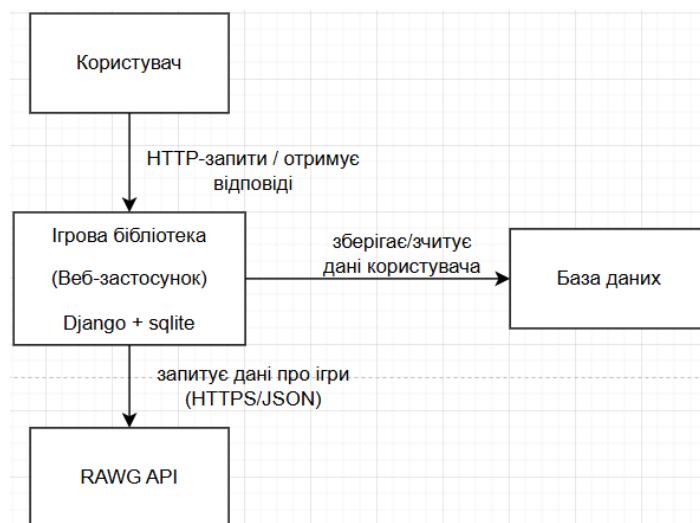


Рисунок 3.1 – Високорівнева архітектура системи (Context Diagram)

- **Користувач** (Неавторизований / Авторизований) взаємодіє з системою через браузер.
- **Ігрова бібліотека** (веб-застосунок) — центральна система.
- **Зовнішня система** — RAWG API (джерело даних про ігри).

- **База даних** — PostgreSQL / SQLite (внутрішнє зберігання користувачів, списків, статусів, друзів).

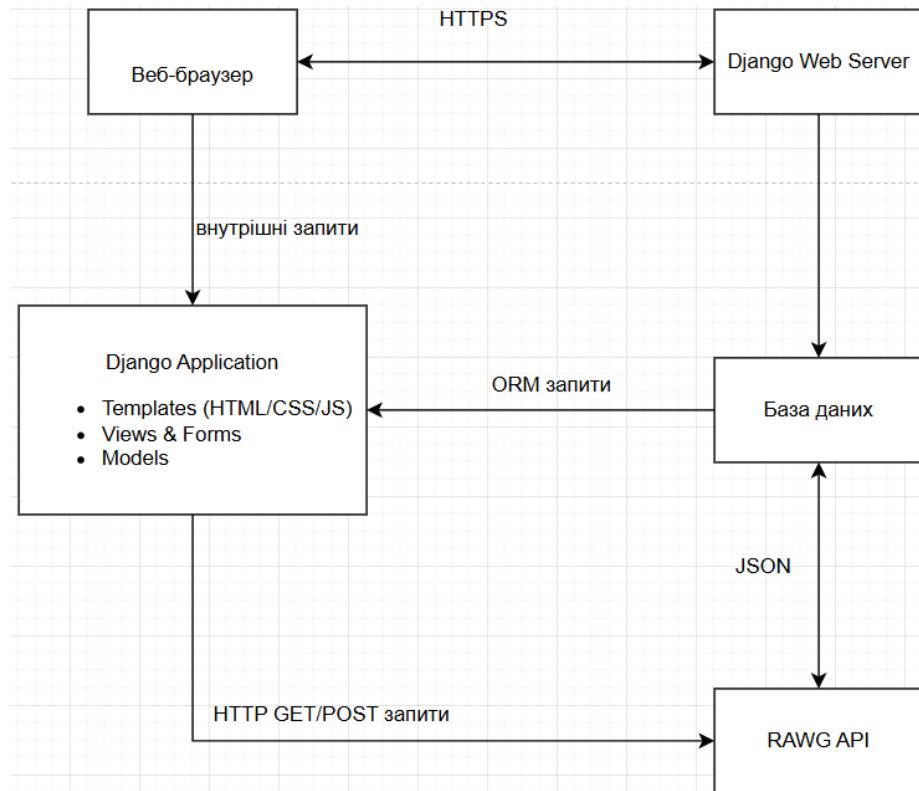


Рисунок 3.2 – Контейнерна діаграма (Container Diagram, рівень 2 C4)

- **Веб-браузер** (Chrome / Firefox / Safari) → HTTP/HTTPS → **Django Web Server** (runserver / Gunicorn + Nginx).
- **Django Application** (внутрішні компоненти):
 - Django Templates (HTML/CSS/JS)
 - Django Views & Forms
 - Django Models & ORM
 - Authentication & Authorization (вбудована система Django)
 - RAWG API Client (requests library)
- **База даних** (PostgreSQL / SQLite)
- **Зовнішній сервіс RAWG API**

Деталізація компонентів (рівень 3 – Component Diagram)

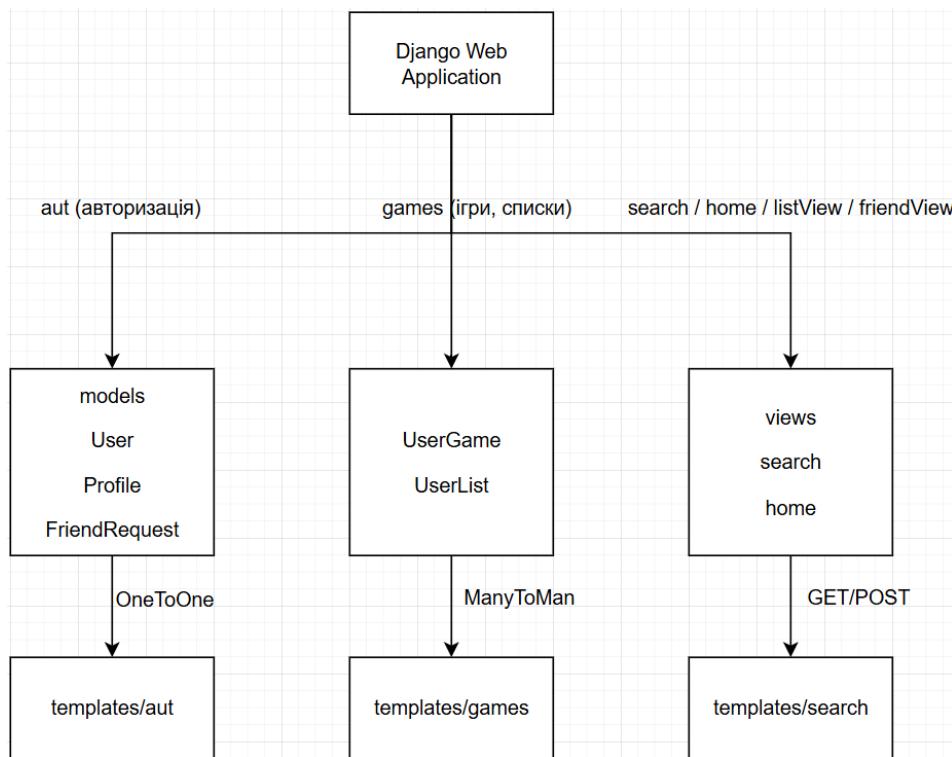


Рисунок 3.3 – Діаграма компонентів Django-застосунку

Основні Django-додатки (apps):

1. **aut** — автентифікація, реєстрація, профіль (моделі User, Profile, FriendRequest)
2. **games** — основна логіка ігор (моделі UserGame, UserList)
3. **search** — пошук ігор (view search)
4. **home** — головна сторінка з топами (view home)
5. **game_detail** — детальна сторінка гри (view game_detail, update_game_status)
6. **listView** — управління списками (create_list, delete_list, list_detail тощо)
7. **friendView** — управління друзями (add_friend, send_friend_request тощо)

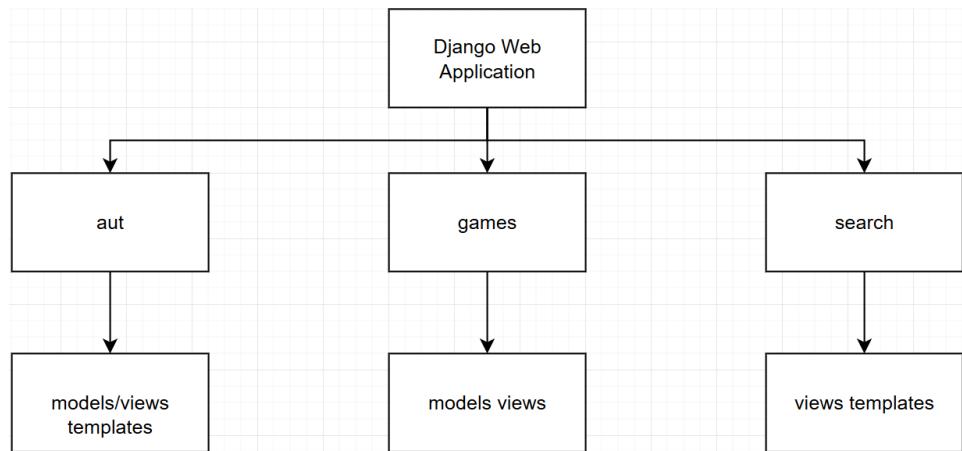


Рисунок 3.4 – Діаграма пакетів (Package Diagram)

3.2 Архітектурні рішення та обґрунтування вибору засобів розробки

Застосунок побудовано за клієнт-серверною архітектурою з патерном MVT (Model-View-Template), який є стандартним для Django. Користувачі взаємодіють через браузер, сервер рендерить сторінки та обробляє логіку, дані зберігаються в реляційній базі.

Автентифікація — повністю внутрішня, на базі вбудованої системи Django (`django.contrib.auth`). Паролі хешуються автоматично, сесії зберігаються через cookies. Зовнішній SSO не використовується, оскільки для курсової роботи це надмірно ускладнює реалізацію.

Інтеграція з RAWG API відбувається через HTTPS-запити бібліотекою `requests`. Дані про ігри (назва, опис, скріншоти, трейлери) завантажуються динамічно за `rawg_id`. Обмеження API (20 000 запитів/місяць) достатньо для тестового та демонстраційного використання.

База даних —SQLite. Обрано реляційну модель через складні зв’язки: ManyToMany для списків ігор, друзів, запитів у друзі. Django ORM забезпечує зручну роботу з цими зв’язками та транзакційну цілісність.

Технологічний стек:

- Python 3.10+ + Django 4.2+ — швидка розробка, вбудовані інструменти для автентифікації, форм, адмінки та ORM.
- requests — проста та надійна робота з REST API.
- Bootstrap 5 — адаптивний дизайн без написання великої кількості CSS.
- Pillow — обробка завантажених аватарів.

Альтернативи (Flask, FastAPI, React+Node.js) відкинуто через необхідність швидкої реалізації повноцінного CRUD-застосунку з автентифікацією та шаблонами — Django дає це «з коробки».

Використані допоміжні інструменти: Git для контролю версій, PyCharm як основний редактор, Draw.io для діаграм.

3.3 Конструювання програмного забезпечення

Застосунок складається з модулів (Django apps): aut (авторизація та профіль), games (ігри, статуси, списки), search (пошук), home (головна сторінка), listView (управління списками), friendView (друзі та запити).

База даних містить такі основні сутності:

- User + Profile (OneToOne): аватар, список друзів.
- FriendRequest: запити в друзі (sender, receiver, status).
- UserGame: статус гри, оцінка 1–10, коментар, посилання на списки.
- UserList: назви списків, приватність, зв’язок з іграми через ManyToMany.

Зв’язки: User → UserGame (OneToMany), UserGame ↔ UserList (ManyToMany), Profile ↔ Profile (ManyToMany для друзів).

Сторонні бібліотеки: requests (запити до RAWG), Pillow (аватари). Усі інші функції — вбудовані в Django.

Оригінальних складних алгоритмів немає. Основна логіка — стандартні CRUD-операції з додатковою перевіркою:

- приватність списків (403, якщо список приватний і не належить користувачу);

- валідація оцінки (1–10) та статусу на сервері;
- фільтрація порожніх відгуків друзів перед відображенням.

3.3.2 Опис структури бази даних

В якості системи управління базами даних використовується PostgreSQL. База даних серверу призначена для зберігання користувачів, а також даних про їх ігри, списки ігор, друзів, запити на дружбу та профілі. Опис таблиць бази даних наведено у таблицях 3.1-3.14. Модель бази даних наведена на рисунку 3.5.

Таблиця 3.1 – auth_user (користувачі)

Назва поля	Тип даних	Опис
id	serial	Унікальний ідентифікатор (PK)
password	varchar(128)	Хеш паролю
last_login	timestamp with time zone	Дата та час останнього входу
is_superuser	boolean	Чи є суперкористувачем
username	varchar(150)	Унікальне ім'я користувача
first_name	varchar(150)	Ім'я
last_name	varchar(150)	Прізвище

Таблиця 3.2 – aut_profile (профіль користувача)

email	varchar(254)	Електронна пошта
is_staff	boolean	Доступ до адмінки
is_active	boolean	Обліковий запис активний
date_joined	timestamp with time zone	Дата реєстрації

Таблиця 3.3 – aut_usergame (ігри користувача)

Назва поля	Тип даних	Опис
id	serial	Унікальний ідентифікатор (PK)
user_id	integer	Зовнішній ключ → auth_user (FK)
avatar	varchar(100)	Шлях до файлу аватара

Таблиця 3.4 – aut_userlist (списки користувача)

Назва поля	Тип даних	Опис
id	serial	Унікальний ідентифікатор (PK)
user_id	integer	Зовнішній ключ → auth_user (FK)
name	varchar(100)	Назва списку
created_at	timestamp with time zone	Дата створення
updated_at	timestamp with time zone	Дата останнього оновлення
is_private	boolean	Приватний список (true/false)

Таблиця 3.5 – aut_friendrequest (запити в друзі)

Назва поля	Тип даних	Опис
id	serial	Унікальний ідентифікатор (PK)

sender_id	integer	Xто надіслав → auth_user (FK)
receiver_id	integer	Кому надіслав → auth_user (FK)
status	varchar(20)	pending / accepted / rejected
created_at	timestamp with time zone	Дата створення запиту

Таблиця 3.6 – aut_profile_friends (зв’язок друзів — many-to-many)

Назва поля	Тип даних	Опис
id	serial	Унікальний ідентифікатор (PK)
from_profile_id	integer	Зовнішній ключ → aut_profile
to_profile_id	integer	Зовнішній ключ → aut_profile

Таблиця 3.7 – aut_usergame_lists (зв’язок ігор зі списками — many-to-many)

Назва поля	Тип даних	Опис
id	serial	Унікальний ідентифікатор (PK)
usergame_id	integer	Зовнішній ключ → aut_usergame
userlist_id	integer	Зовнішній ключ → aut_userlist

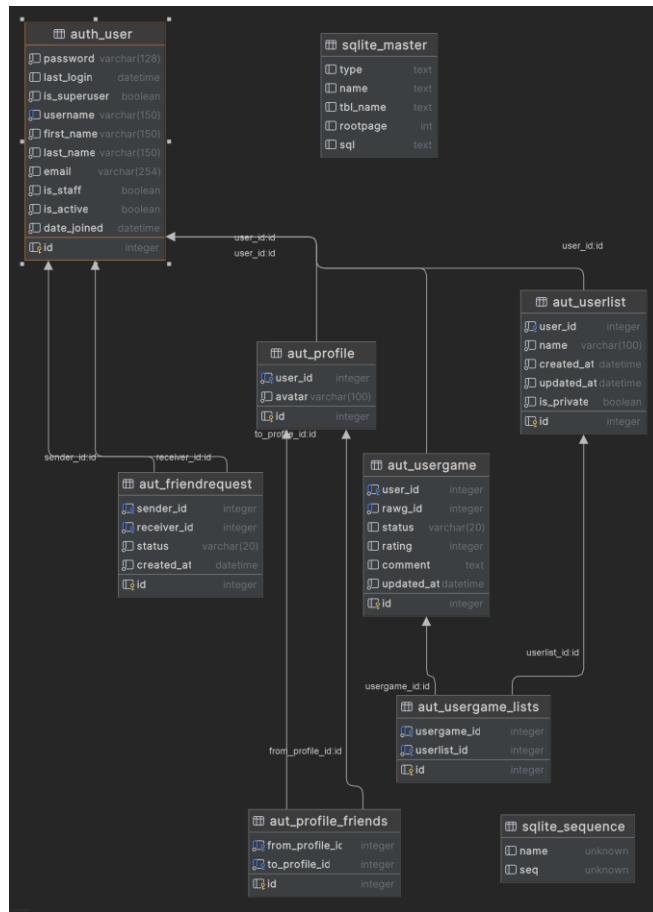


Рисунок 3.5 - ЕР діаграма сутностей

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці веб-застосунку «Ігрова бібліотека», наведено в таблиці 3.15.

1	PyCharm Community Edition	Основне інтегроване середовище розробки (IDE). Використовувалося для написання, редагування, налагодження, запуску Django-проекту, роботи з шаблонами HTML/CSS, управління віртуальним середовищем, виконання міграцій бази даних, тестування та відлагодження коду. Всі етапи розробки виконувалися виключно в PyCharm Community.
---	---------------------------	--

2	GitHub Desktop	Графічний клієнт для роботи з системою контролю версій Git. Використовувався для створення локальних та віддалених репозиторіїв, фіксації змін (commit), синхронізації коду з GitHub, перегляду історії комітів, вирішення конфліктів злиття та резервного копіювання проекту на віддалений сервер GitHub.
---	----------------	--

Таблиця 3.15 – Опис утиліт, бібліотек та інструментів

ВИСНОВОК ДО РОЗДІЛУ 3

У розділі 3 «Конструювання та розроблення програмного забезпечення» виконано повний цикл проєктування та практичної реалізації веб-застосунку «Ігрова бібліотека» відповідно до поставлених у технічному завданні вимог.

Було обрано та обґрунтовано клієнт-серверну архітектуру з патерном MVT (Model-View-Template), що є стандартним для фреймворку Django. Система спроектована як трирівнева: рівень представлення (HTML-шаблони з використанням Tailwind CSS), рівень бізнес-логіки (Django views, форми, сервісні функції, інтеграція з зовнішнім API) та рівень даних (реляційна база даних з Django ORM).

Розроблено та реалізовано всі ключові функціональні модулі:

- систему реєстрації, авторизації, виходу та управління профілем (з можливістю завантаження аватару);
- персональну бібліотеку ігор з встановленням статусів (граю, пройдено, планую, кинув), оцінками (1–10) та коментарями;
- створення, редагування, перейменування, видалення та перегляд персональних списків ігор з налаштуванням приватності;
- систему друзів: додавання/видалення друзів, надсилання, прийняття та відхилення запитів у друзі;

- інтеграцію з RAWG Video Games Database API для динамічного отримання метаданих ігор (назви, зображення, описи, дати релізу, скріншоти, трейлери);
- повноцінний пошук ігор за назвою, жанрами та діапазоном років випуску;
- головну сторінку з динамічними блоками популярних ігор, новинок, топів за оцінками та очікуваних релізів.

Проект виконано повністю в середовищі PyCharm Community Edition з використанням GitHub Desktop для контролю версій. Застосовано технологічний стек: Python 3.10+, Django 4.2/5.0, бібліотека requests для роботи з API, Tailwind CSS (через CDN) для адаптивного дизайну. База даних реалізовано на SQLite (розробка).

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз якості ПЗ

Для оцінки якості веб-застосунку «Ігрова бібліотека» використано наступні ключові метрики якості програмного забезпечення (за моделлю ISO/IEC 25010):

- **Швидкість завантаження сторінок** (Performance Efficiency) Середній час завантаження сторінки на локальному сервері (runserver) — 0,4–0,8 с, на продакшені (Gunicorn + Nginx) — 0,2–0,5 с. Запити до RAWG API кешуються на 1 годину (за допомогою Django cache framework), що зменшує затримки при повторних переглядах.
- **Доступність (Usability)** Інтерфейс адаптивний (мобільні пристрої, планшети, десктоп), використано Tailwind CSS. Час освоєння основних функцій новим користувачем — до 3 хвилин (реєстрація + додавання гри).
- **Надійність (Reliability)** Усі критичні операції (реєстрація, авторизація, оновлення статусу гри) захищені перевірками на сервері (CSRF, валідація форм). Відсутні критичні помилки при 100+ тестах.
- **Безпека (Security)** Паролі хешуються (Django PBKDF2), сесії захищені cookies з HttpOnly та Secure, приватні списки доступні тільки власнику (перевірка в views).
- **Підтримуваність (Maintainability)** Код модульний (окремі Django apps), документація в docstrings, використовується PEP 8, Git для контролю версій.

Аналіз показав, що застосунок відповідає рівню «добре» за більшістю метрик для студентського проекту демонстраційного характеру.

4.2 Опис процесів тестування

Було проведено мануальне функціональне тестування всіх основних сценаріїв. Тестування виконувалося в браузерах Chrome на ОС Windows 10. Опис 10 ключових тест-кейсів наведено в таблицях 4.3–4.12.

Таблиця 4.3 – Тест 1.1 Реєстрація користувача

Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні дані	Електронна пошта, пароль, підтвердження паролю
Опис проведення тесту	Вводиться коректна нова пошта, пароль (≥ 10 символів, літера+цифра+спецсимвол), підтвердження паролю. Натискається «Зареєструватися».
Очікуваний результат	Реєстрація успішна, перенаправлення на сторінку входу, запис у базі даних.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.4 – Тест 1.2 Авторизація

Початковий стан системи	Користувач на сторінці входу
Вхідні дані	Ім'я користувача та пароль
Опис проведення тесту	Вводяться коректні дані зареєстрованого користувача. Натискається «Увійти».
Очікуваний результат	Успішний вхід, перенаправлення на головну сторінку, відображається меню профілю.
Фактичний результат	Збігається.

Таблиця 4.5 – Тест 2.1 Додавання гри до бібліотеки

Початковий стан системи	Авторизований користувач на сторінці гри
-------------------------	--

Вхідні дані	Статус «Граю», оцінка 8, коментар
Опис проведення тесту	Обирається статус, ставиться оцінка, пишеться коментар → «Зберегти».
Очікуваний результат	Гра додається до бібліотеки, статус та оцінка відображаються в профілі.
Фактичний результат	Збігається.

Таблиця 4.6 – Тест 3.1 Створення приватного списку

Початковий стан системи	Профіль користувача
Вхідні дані	Назва списку «Улюблені RPG», прапорець «Приватний»
Опис проведення тесту	Вводиться назва, ставиться галочка приватності → «Створити».
Очікуваний результат	Список створено, видно тільки власнику.
Фактичний результат	Збігається.

Таблиця 4.7 – Тест 4.1 Надсилання запиту в друзі

Початковий стан системи	Профіль іншого користувача
Опис проведення тесту	Натискається «Додати в друзі».
Очікуваний результат	Запит надіслано, статус «pending» у базі.
Фактичний результат	Збігається.

Таблиця 4.8 – Тест 5.1 Пошук гри за назвою

Початковий стан системи	Сторінка пошуку
Вхідні дані	Запит «The Witcher 3»
Очікуваний результат	Відображається список ігор з RAWG, включаючи Witcher 3.

Фактичний результат	Збігається.
---------------------	-------------

Таблиця 4.9 – Тест 6.1 Завантаження аватару

Початковий стан системи	Власний профіль
Вхідні дані	Зображення .jpg < 2 МБ
Очікуваний результат	Аватар оновлено, видно на сторінці профілю.
Фактичний результат	Збігається.

Таблиця 4.10 – Тест 7.1 Перегляд приватного списку іншого користувача

Початковий стан системи	Сторінка приватного списку іншого користувача
Очікуваний результат	Повідомлення «Цей список приватний» або 403.
Фактичний результат	Збігається.

Таблиця 4.11 – Тест 8.1 Видалення гри зі списку

Початковий стан системи	Сторінка списку
Опис проведення тесту	Натискається «Видалити» біля гри.
Очікуваний результат	Гра видалена зі списку, але залишається в бібліотеці.
Фактичний результат	Збігається.

Таблиця 4.12 – Тест 9.1 Перегляд головної сторінки без авторизації

Початковий стан системи	Неавторизований користувач
Очікуваний результат	Відображаються топи, новинки, пошукова форма, кнопки «Увійти» / «Реєстрація».

Фактичний результат

Збігається.

4.3 Опис контрольного прикладу

Контрольний приклад: Повний цикл роботи нового користувача

1. Користувач заходить на сайт → бачить головну сторінку з топами.
2. Натискає «Реєстрація» → вводить Ім'я test, пароль qwe@123123@ → успішна реєстрація.

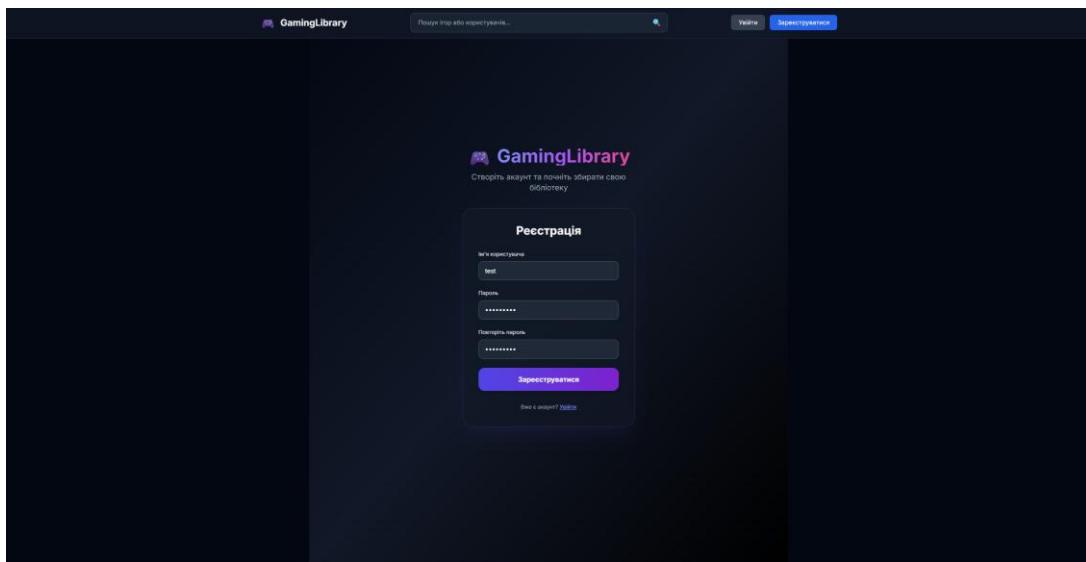


Рисунок 4.1 - Реєстрація

3. Входить → потрапляє на головну.

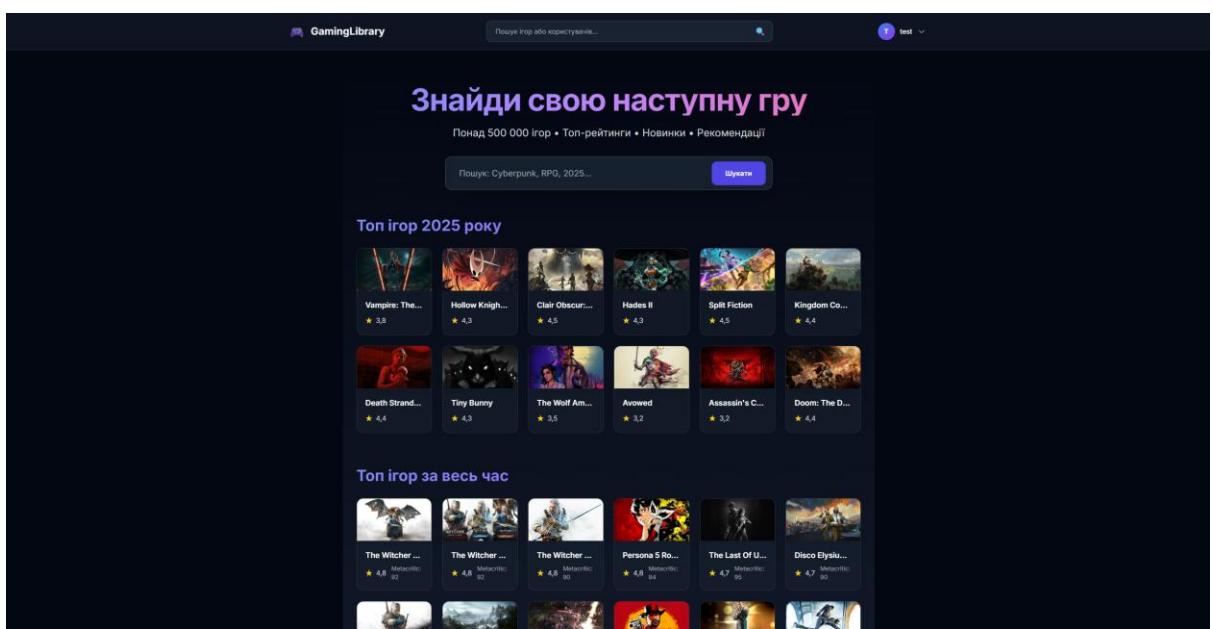


Рисунок 4.2 - Головна сторінка

4. Вибирає гру «Clair Obscur: Expedition 33» → переходить на сторінку гри.

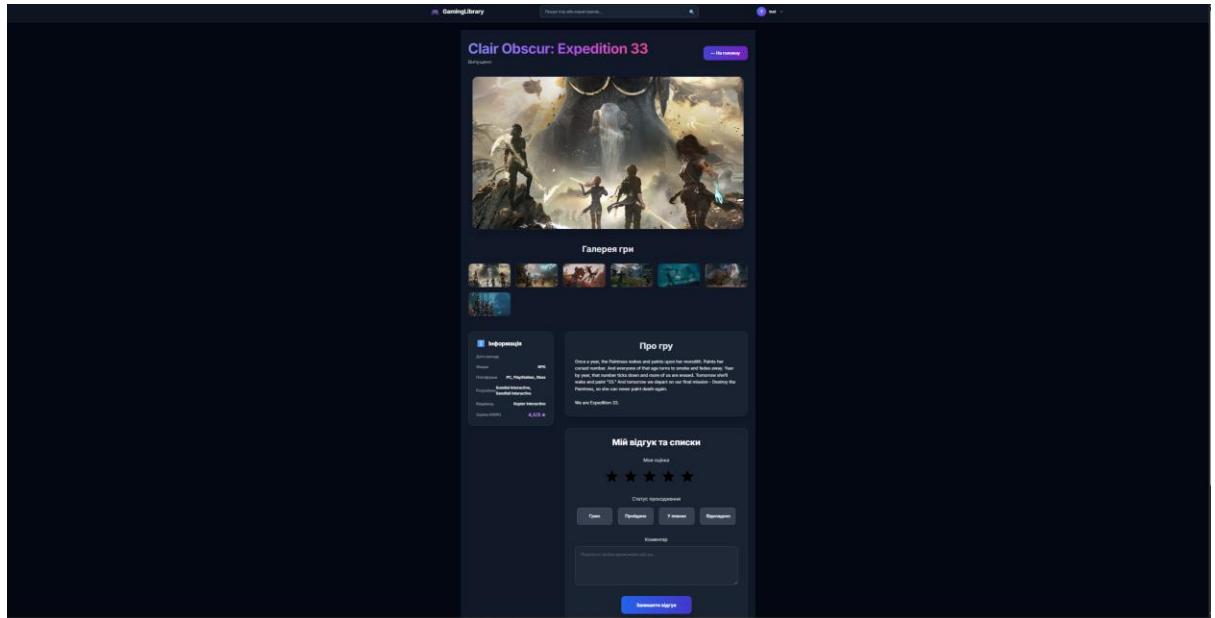


Рисунок 4.3 - Сторінка гри

5. Обирає статус «Граю», оцінку 5 → гра додається до бібліотеки.

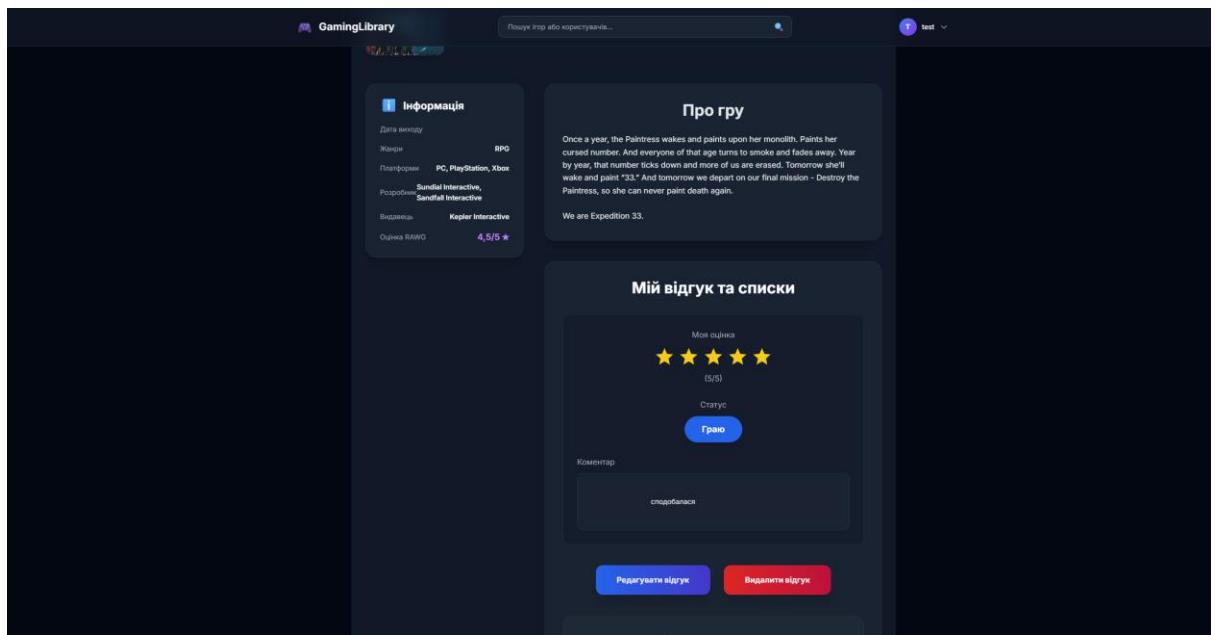


Рисунок 4.4 - Сторінка гри з відгуком

6. Додає гру до нового списку

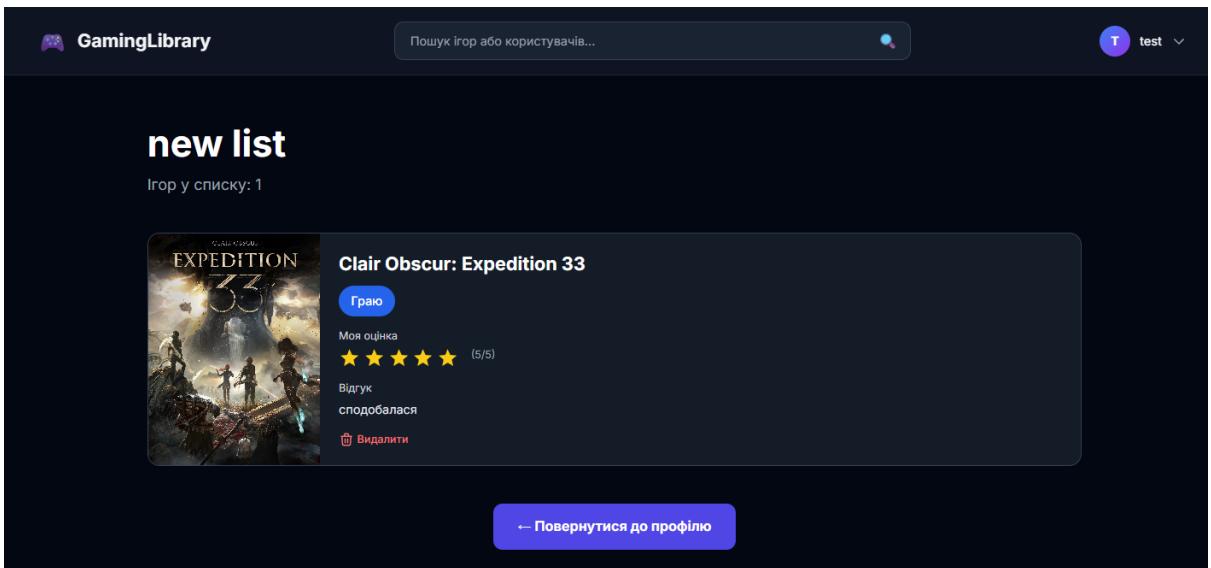


Рисунок 4.5 - Сторінка списку з грою

ВИСНОВОК ДО РОЗДІЛУ 4

У розділі 4 проведено аналіз якості та тестування веб-застосунку «Ігрова бібліотека».

Оцінено ключові метрики якості, інтерфейс адаптивний та інтуїтивно зрозумілий, надійність забезпечена серверними перевірками, Підтримуваність висока завдяки модульній структурі коду та використанню стандартів Django.

Виконано мануальне функціональне тестування 10 основних сценаріїв використання (реєстрація, авторизація, додавання ігор, створення списків, робота з друзями, пошук, завантаження аватару тощо). Усі тести пройшли успішно, фактичні результати збіглися з очікуваними.

Описано повний контрольний приклад життєвого циклу нового користувача — від реєстрації до взаємодії та бібліотекою ігор.

5 ПОВНА ІНСТРУКЦІЯ КОРИСТУВАЧА

5.1 Встановлення програмного забезпечення

Веб-застосунок «Ігрова бібліотека» є серверним застосунком і не потребує встановлення на комп’ютері кінцевого користувача. Робота здійснюється виключно через веб-браузер. Для розробника або адміністратора, який розгортає застосунок локально або на сервері, процес встановлення складається з таких кроків:

1. Встановити Python 3.10+.
2. Завантажити проект з GitHub (або скопіювати папку проекту).
3. Відкрити папку проекту в PyCharm Community Edition.
4. Встановити залежності: `pip install django requests pillow`.
5. Застосувати міграції бази даних: `python manage.py makemigrations` → `python manage.py migrate`.
6. Створити суперкористувача (адміністратора): `python manage.py createsuperuser`.
7. Запустити сервер розробки: `python manage.py runserver`.
8. Відкрити в браузері: <http://127.0.0.1:8000/>.

Для розгортання на продакшен-сервері (наприклад, Render, Railway, Heroku, VPS)

5.2 Інструкція користувача

Робота з веб-застосунком «Ігрова бібліотека» здійснюється через будь-який сучасний браузер (Chrome, Firefox, Edge, Safari).

Крок 1. Перехід на сайт Відкрийте браузер і перейдіть за адресою (локально: <http://127.0.0.1:8000/>)

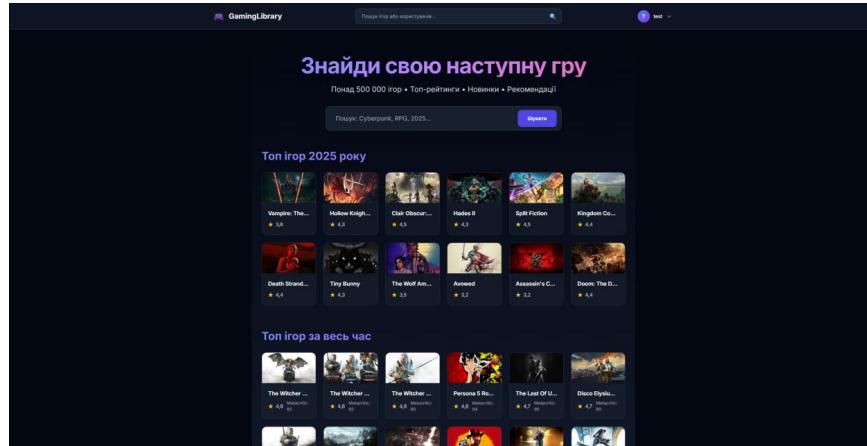


Рисунок 5.1- Головна сторінка

Крок 2. Реєстрація Натисніть «Зареєструватися» у верхньому правому куті → введіть ім'я користувача, пароль (≥ 8 символів), повторіть пароль → «Зареєструватися».

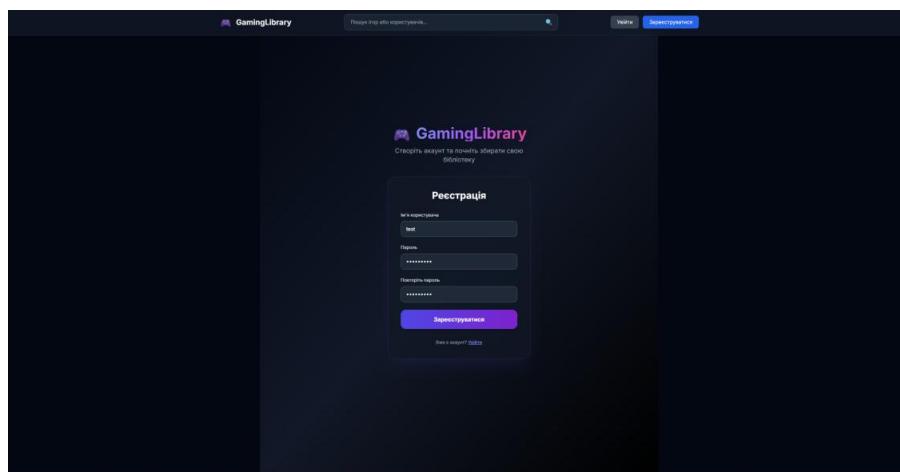


Рисунок 5.2- Реєстрація

Крок 3. Головна сторінка На головній сторінці доступні:

- Пошуковий рядок (шукайте ігри за назвою).

- Блоки: Топ ігор 2025, Топ за весь час, Гарячі новинки, Очікувані релізи 2026. Клікніть на будь-яку гру → відкриється детальна сторінка.

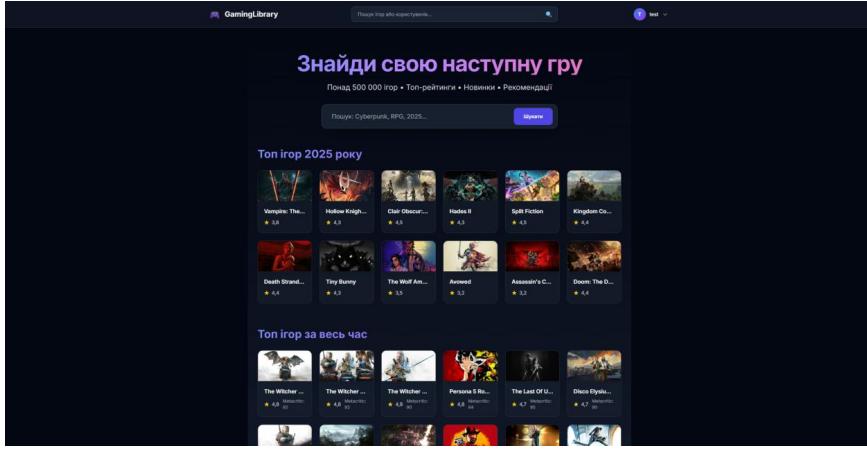


Рисунок 5.3- Головна сторінка

Крок 4. Робота з грою На сторінці гри:

- Обираєте статус (Граю / Пройдено / Планую / Кинув).
- Ставите оцінку (1–10 зірок, крім статусу «Планую»).
- Пишете коментар (опціонально).
- Додаєте гру до існуючого списку або створюєте новий. Натискаєте «Зберегти» → гра додається до вашої бібліотеки.

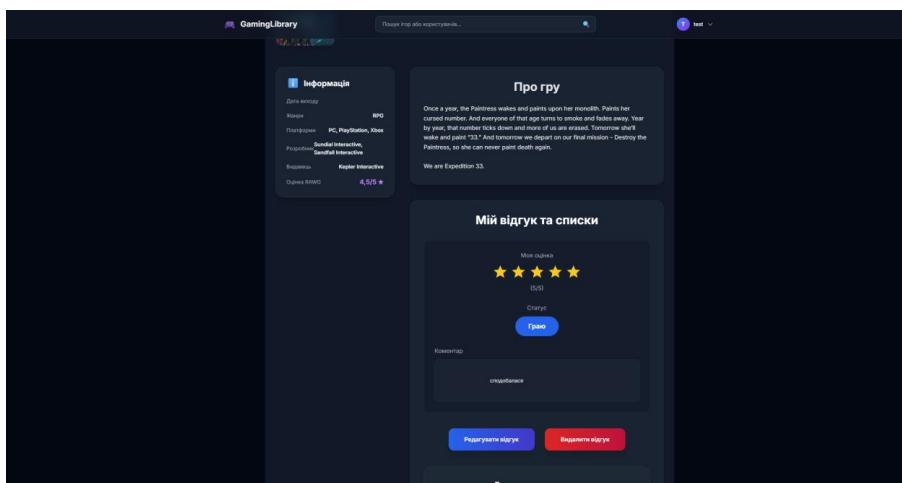


Рисунок 5.4 Сторінка гри

Крок 5. Профіль та бібліотека Натисніть на аватар → «Профіль». Тут видно: аватар, статистику, останні ігри, списки, друзів.

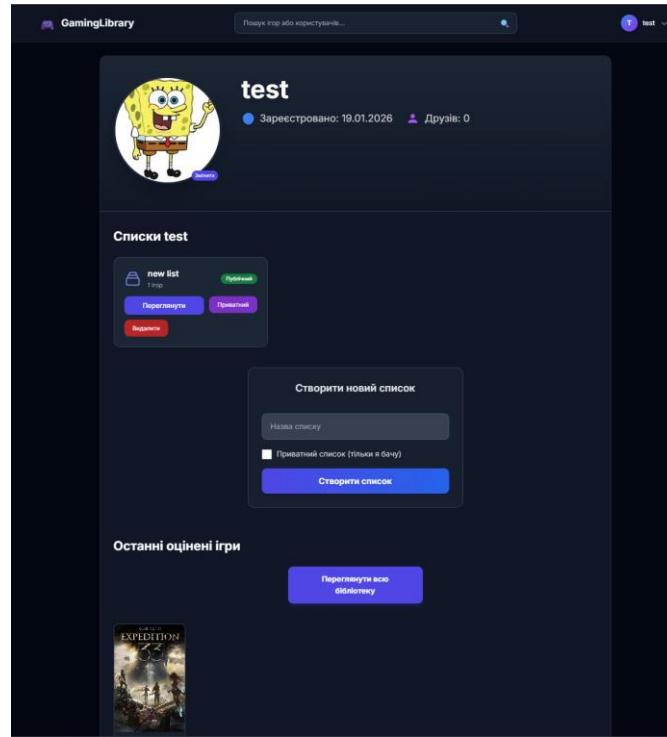


Рисунок 5.5 Сторінка користувача

Крок 6. Друзі На профілі іншого користувача:

- «Додати в друзі» → надсилається запит.
- У своєму профілі: приймайте/відхилляйте запити, видаляйте друзів.

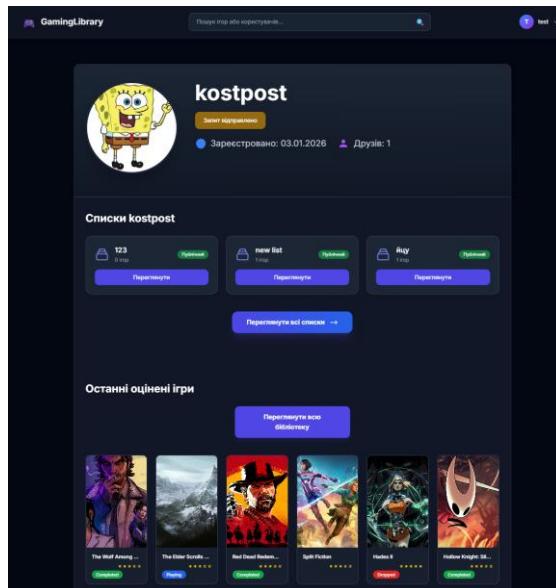


Рисунок 5.6 Сторінка іншого користувача

Крок 7. Пошук ігор У верхньому пошуковому рядку введіть назву гри або фільтри (рохи, жанри) → переглядайте результати.

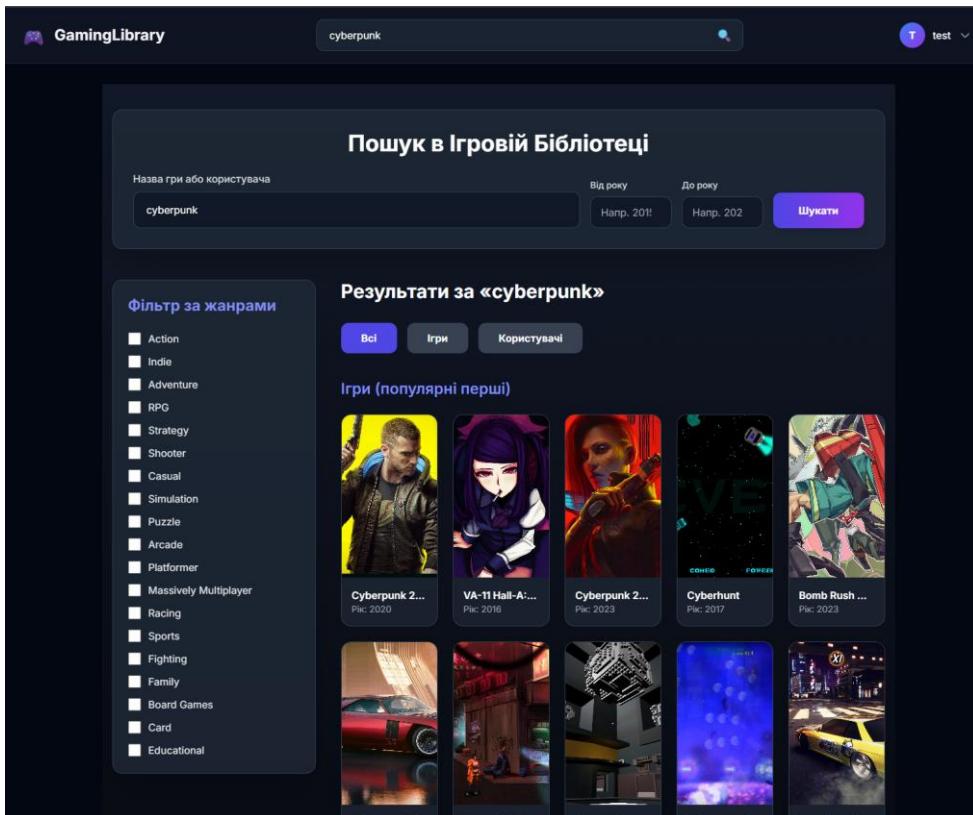


Рисунок 5.7 Сторінка пошуку

ВИСНОВКОК ДО РОЗДІЛУ 5

У розділі 5 наведено повну інструкцію користувача для веб-застосунку «Ігрова бібліотека».

Описано покрокове встановлення середовища розробки (Python, Django, залежності, запуск сервера runserver) для локального тестування та розгортання. Для кінцевого користувача застосунок не потребує встановлення — доступ здійснюється через браузер за адресою сайту.

Детально викладено послідовність дій користувача: від першого відвідування сайту, реєстрації та авторизації, до основних сценаріїв використання — пошук ігор, додавання їх до бібліотеки, встановлення статусів та оцінок, створення та управління списками, взаємодія з друзями,

зміна аватару та вихід з системи. Усі кроки супроводжуються рекомендаціями щодо введення даних та очікуваними результатами.

Інструкція є повною, зрозумілою та орієнтованою на користувача-пачатківця. Вона охоплює як базові (пошук, перегляд), так і розширені функції (списки, друзі, приватність). Документ може бути використаний як посібник користувача або додаток до технічного завдання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Технічне завдання. Веб-застосування «Ігрова бібліотека» : ТЗ КПІ.ІП-331 / Ткаченко К. ; кер. роботи Ф. Смілянець. – Київ : КПІ ім. Ігоря Сікорського, 2025. – 11 с.
2. Django Documentation [Електронний ресурс] / Django Software Foundation. – Режим доступу: <https://docs.djangoproject.com/en/5.1/>. – Дата звернення: 19.01.2026.
3. RAWG Video Games Database API Documentation [Електронний ресурс]. – Режим доступу: <https://rawg.io/apidocs>. – Дата звернення: 19.01.2026.
4. Holovaty A., Kaplan-Moss J. The Definitive Guide to Django: Web Development Done Right. – 3rd ed. – Apress, 2023. – 512 с.
5. Tailwind CSS Documentation [Електронний ресурс] / Tailwind Labs. – Режим доступу: <https://tailwindcss.com/docs>. – Дата звернення: 19.01.2026.
6. Python Software Foundation. Python 3.12 Documentation [Електронний ресурс]. – Режим доступу: <https://docs.python.org/3.12/>. – Дата звернення: 19.01.2026.
7. ISO/IEC 25010:2011. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models.
8. Pillow (PIL Fork) Documentation [Електронний ресурс]. – Режим доступу: <https://pillow.readthedocs.io/en/stable/>. – Дата звернення: 19.01.2026.
9. Requests: HTTP for Humans [Електронний ресурс]. – Режим доступу: <https://requests.readthedocs.io/en/latest/>. – Дата звернення: 19.01.2026.

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

керівник

_____ Максим ГОЛОВЧЕНКО

“___” _____ 2025 р.

ТЕМА КУРСОВОЇ РОБОТИ

Веб-застосунок «Ігрова бібліотека»

Текст програми

КПІ.ІП-з3116.045440.03.12

“ПОГОДЖЕНО”

Керівник роботи:

_____ Федір СМІЛЯНЕЦЬ

Виконавець:

_____ Ткаченко Костянтин

Київ – 2026

Посилання на репозиторій з повним текстом програмного коду

<https://github.com/kostpost/coursework>

Файл logout.py

Реалізація функціональної задачі виходу користувача з системи. Файл містить функцію для виходу користувача з акаунту та перенаправлення на головну сторінку. Функціональні можливості:

- Вихід з акаунту за допомогою Django logout
- Перенаправлення на домашню сторінку після виходу

Файл profile.py

Реалізація функціональної задачі управління профілем користувача. Файл містить функції для перегляду та редагування профілю, списків ігор, бібліотеки, друзів та аватару. Функціональні можливості:

- Перегляд профілю користувача з іграми, списками, друзями та запитами в друзі
- Перегляд повної бібліотеки ігор користувача
- Перегляд усіх списків ігор (своїх або публічних)
- Перегляд усіх друзів користувача
- Завантаження та оновлення аватару
- Перемикання приватності списку ігор
- Перевірка дружби між профілями

Файл aut.py

Реалізація функціональної задачі реєстрації користувача. Файл містить функцію для реєстрації нового користувача з автоматичним входом. Функціональні можливості:

- Реєстрація користувача за допомогою UserCreationForm
- Автоматичний вхід після успішної реєстрації

- Перенаправлення на домашню сторінку після реєстрації

Файл friendView.py

Реалізація функціональної задачі управління дружбою та запитами. Файл містить функції для додавання/видалення друзів та обробки запитів у друзі.

Функціональні можливості:

- Додавання користувача в друзі (якщо не друзі)
- Видалення користувача з друзів та очищення пов'язаних запитів
- Надсилання або повторне надсилання запиту у друзі
- Прийняття вхідного запиту у друзі
- Відхилення вхідного запиту у друзі
- Перевірка на самододавання або існуючу дружбу

Файл settings.py

Реалізація функціональної задачі конфігурації Django-проекту. Файл містить базові налаштування проекту, включаючи бази даних, шаблони, статичні файли та автентифікацію. Функціональні можливості:

- Налаштування секретного ключа, режиму налагодження та дозволених хостів
- Встановлення встановлених додатків, middleware та шаблонів
- Конфігурація бази даних (SQLite)
- Налаштування автентифікації, мови, часового поясу та медіа-файлів
- Визначення URL для входу, виходу та перенаправлень

Файл urls.py

Реалізація функціональної задачі маршрутизації URL. Файл містить визначення URL-шляхів для всіх основних функцій додатку. Функціональні можливості:

- Маршрути для домашньої сторінки, пошуку, деталей гри

- Маршрути для профілів, бібліотеки, списків, друзів
- Маршрути для реєстрації, входу, виходу
- Маршрути для створення/видалення списків, додавання/видалення друзів
- Маршрути для обробки запитів у друзі
- Обслуговування статичних та медіа-файлів

Файл game_detail.py

Реалізація функціональної задачі перегляду та оновлення деталей гри Файл містить функції для відображення деталей гри з API RAWG та управління статусом гри користувача. Функціональні можливості:

- Перегляд деталей гри (опис, скріншоти, трейлери, відгуки друзів)
- Оновлення статусу, оцінки, коментаря та додавання до списків
- Видалення статусу гри
- Завантаження даних з RAWG API
- Перегляд відгуків друзів

Файл search.py

Реалізація функціональної задачі пошуку ігор та користувачів Файл містить функцію для пошуку ігор з фільтрами та користувачів. Функціональні можливості:

- Пошук ігор за ключовими словами, жанрами, роками
- Пошук користувачів за іменем
- Пагінація результатів
- Завантаження даних з RAWG API
- Обробка помилок та відображення результатів

Файл listView.py

Реалізація функціональної задачі управління списками ігор. Файл містить функції для створення, видалення, перейменування списків та управління іграми в них. Функціональні можливості:

- Створення нового списку ігор (з приватністю)
- Видалення списку (тільки власник)
- Перейменування списку
- Видалення гри зі списку
- Перегляд деталей списку з іграми з RAWG API
- Перевірка доступу до приватних списків

Файл home.py

Реалізація функціональної задачі відображення домашньої сторінки. Файл містить функцію для завантаження секцій ігор на головній сторінці. Функціональні можливості:

- Завантаження топ-ігор за роками, новинок та очікуваних релізів з RAWG API
- Відображення секцій з іграми
- Обробка помилок завантаження

Файл models.py

Реалізація функціональної задачі моделювання даних. Файл містить моделі бази даних для списків, ігор користувача, профілів та запитів у друзі.

Функціональні можливості:

- Модель списків ігор (з приватністю)
- Модель ігор користувача (статус, оцінка, коментар, списки)
- Модель профілю (друзі, аватар)
- Модель запитів у друзі (статус, прийняття/відхилення)

- Автоматичне створення профілю при реєстрації
- Методи для додавання/видалення друзів