МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп’ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

УТЮК Антон Романович

Веб-орієнтована інформаційна система генерування контенту для настільних рольових ігор / Web-oriented information system for generating content for tabletop role-playing games

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Комп’ютерні науки

Кваліфікаційна робота

Виконав студент групи КН-42
А. Р. Утюк
_____

Науковий керівник:
к.т.н., доцент І. В. Турченко
_____

Кваліфікаційну роботу
допущено до захисту:
«___» _____ 20___ р.
Завідувач кафедри
_____ Н.В. Дзюбановська

ТЕРНОПІЛЬ – 2026

Факультет комп’ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність: 122 – Комп’ютерні науки
освітньо-професійна програма – Комп’ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.В. Дзюбановська
«____» _____20__р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
УТЮКУ Антону Романовичу
(прізвище, ім’я, по батькові)

1. Тема кваліфікаційної роботи

Веб-орієнтована інформаційна система генерування контенту для настільних рольових ігор / Web-oriented information system for generating content for tabletop role-playing games

керівник роботи к.т.н., доцент І.В. Турченко

затверджений наказом по університету від 1 грудня 2025 року № 894.

2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2026 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити

- провести аналіз предметної області та огляд існуючих веб-орієнтованих інформаційних систем генерування контенту для настільно-рольових ігор, виявити їхні переваги та недоліки;
- дослідити особливості автоматизації процесу генерування ігрового контенту та взаємодії користувача із системою;
- спроєктувати структуру веб-орієнтованої інформаційної системи генерування контенту для настільних рольових ігор;
-спроєктувати інформаційне та інформаційне забезпечення системи;
- реалізувати веб-орієнтовану інформаційну систему генерування контенту для настільних рольових ігор;
- провести тестування програмної системи та оцінити коректність її роботи.

5. Перелік графічного матеріалу у роботі

- Схема алгоритму генерування ігрового контенту

– UML-діаграма класів

6. Консультанти розділів кваліфікаційної роботи

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

7. Дата видачі завдання 01 грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Огляд літературних джерел відповідно до предметної області та складання плану роботи	до 01.01.2026 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.02.2026 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 15.03.2026 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05.2026 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2026 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2026 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2026 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2026 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06.2026 р.	

Студент _____ А.Р. Утюк

підпис

Керівник роботи _____ к.т.н., доцент І.В. Турченко

підпис

АНОТАЦІЯ

Кваліфікаційна робота на тему «Веб-орієнтована інформаційна система генерування контенту для настільних рольових ігор» на здобуття освітнього ступеня «Бакалавр» зі спеціальності 122 «Комп’ютерні науки» освітньої програми «Комп’ютерні науки» виконана обсягом 55 сторінок і містить 10 рисунків, 6 таблиць, 2 додатки та список з 38 використаних джерел.

Метою кваліфікаційної роботи є проєктування і реалізація веб-орієнтованої інформаційної системи генерування контенту для настільних рольових ігор із застосуванням моделі штучного інтелекту OpenAI GPT-4 та сучасного технологічного стеку.

Об'єктом дослідження є процеси отримання, зберігання, обробки, передачі та використання інформації в системах автоматизованого генерування текстового контенту для настільно-рольових ігор.

Результатом роботи є функціональна веб-орієнтована ІС з клієнт-серверною архітектурою, що включає модуль генерування контенту на базі OpenAI GPT-4, систему збереження результатів у MongoDB та інтерфейс на React.js. Система підтримує генерацію шести типів ігрового контенту: неігрових персонажів (NPC), локацій, таверн, підземель, квестів та артефактів.

Практичне значення отриманих результатів полягає у розробці веб-орієнтованої інформаційної системи генерування контенту для настільних рольових ігор, що забезпечує автоматизацію підготовки ігрових матеріалів засобами штучного інтелекту.

Ключові слова: ВЕБ-ОРІЄНТОВАНА ІНФОРМАЦІЙНА СИСТЕМА, НАСТІЛЬНО-РОЛЬОВІ ІГРИ, ГЕНЕРУВАННЯ КОНТЕНТУ, ВЕЛИКІ МОВНІ МОДЕЛІ, GPT-4, REACT.JS, NODE.JS, MONGODB, REST API, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА.

ANNOTATION

The qualification work entitled “Web-oriented information system for generating content for tabletop role-playing games” for the Bachelor’s degree in Computer Science (specialty 122 “Computer Science”, educational program “Computer Science”) comprises 55 pages and includes 10 figures, 6 tables, 2 appendices, and a list of 38 references.

The purpose of the qualification work is to design and implement a web-oriented information system for generating content for tabletop role-playing games using artificial intelligence technologies (OpenAI GPT-4) and a modern technology stack.

The object of research is the processes of acquisition, storage, processing, transmission and use of information in automated content generation systems for tabletop role-playing games.

The result of the work is a functional web-oriented information system with a client-server architecture, including a content generation module based on OpenAI GPT-4, a result storage system in MongoDB, and a React.js interface. The system supports the generation of six types of game content: non-player characters (NPCs), locations, taverns, dungeons, quests, and artifacts.

The practical significance of the obtained results lies in the development of a web-oriented information system for generating content for tabletop role-playing games, which ensures the automation of game material preparation using artificial intelligence tools.

Keywords: WEB-ORIENTED INFORMATION SYSTEM, TABLETOP ROLE-PLAYING GAMES, CONTENT GENERATION, LARGE LANGUAGE MODELS, GPT-4, REACT.JS, NODE.JS, MONGODB, REST API, CLIENT-SERVER ARCHITECTURE.

ЗМІСТ

Вступ 8
1 Аналіз предметної області та постановка задачі проектування 10
1.1 Опис предметної області процесів генерації контенту настільних рольових ігор 10
1.2 Огляд і аналіз існуючих рішень 15
1.3 Постановка задачі проектування 19
2 Алгоритмічне та інформаційне забезпечення веб-орієнтованої інформаційної системи генерування контенту настільно-рольових ігор 21
2.1 Загальна структура веб-орієнтованої ІС 21
2.2 Алгоритмічне забезпечення 24
2.3 Інформаційне забезпечення 27
3 Реалізація веб-орієнтованої інформаційної системи генерування контенту настільно-рольових ігор 31
3.1 Реалізація програмного забезпечення 31
3.2 Інтерфейс користувача 34
3.3 Тестування програмного забезпечення 36
Висновки 40
Список використаних джерел 41
Додаток А Програмний код 44
Додаток Б Копія опублікованих результатів 48

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

НРІ – настільно-рольові ігри

МГ – майстер гри

ІС – інформаційна система

LLM – Large Language Models

NPC –non-player character

API – Application Programming Interface

REST – Representational State Transfer

SPA – Single Page Application

JWT – JSON Web Token

PCG – Procedural Content Generation

SSE – Server-Sent Events

ODM – Object Document Mapper

RBAC – Role-Based Access Control

FSD – Feature-Sliced Design

СУБД – система управління базами даних

ПЗ – програмне забезпечення

ВСТУП

Індустрія настільно-рольових ігор (HPI) – один із найдавніших і водночас найбільш динамічних жанрів інтерактивних розваг. Від класичних ігор Dungeons & Dragons, уперше виданих у 1974 році, до сучасних систем Pathfinder та Blades in the Dark – індустрія HPI охоплює мільйони гравців по всьому світу. Ключовою фігурою в настільно-рольовій грі є майстер гри (МГ) – людина, яка відповідає за побудову ігрового світу, створення персонажів, локацій, сюжетних ліній та ситуативних описів. Підготовка якісної ігрової сесії вимагає від майстра гри значних часових та творчих ресурсів.

Актуальність дослідження визначається стрімким зростанням інтересу до HPI в Україні та у світі, а також хронічною нестачею інструментів для автоматизованої підтримки майстра гри. За даними аналітиків ICv2 та Circana, обсяг ринку настільно-рольових ігор перевищив 1 млрд доларів США у 2023 році, демонструючи стабільне зростання на рівні 10–15% щорічно. Попри це, більшість МГ досі створюють ігровий контент вручну – описи таверн, характеристики NPC, квестові завдання, карти локацій тощо.

Поява великих мовних моделей (Large Language Models, LLM), зокрема GPT-4 від OpenAI [4], відкрила принципово нові можливості для генерації структурованого текстового контенту. Ці моделі здатні виробляти зв'язний, стилістично витончений та логічно узгоджений текст за мінімальних вхідних даних. Застосування LLM у контексті HPI дозволяє автоматизувати рутинні завдання МГ і суттєво знизити часові витрати на підготовку ігрових матеріалів.

Метою кваліфікаційної роботи є проєктування та реалізація веб-орієнтованої інформаційної системи генерування контенту для настільних рольових ігор із застосуванням OpenAI GPT-4 API та сучасного технологічного стеку: React.js на стороні клієнта, Node.js – на стороні сервера, MongoDB – як система зберігання даних.

Для досягнення визначеної мети необхідно вирішити такі завдання:

– проаналізувати предметну область HPI;

– виконати огляд існуючих програмних рішень, визначити їхні переваги та недоліки;

– спроєктувати архітектуру системи з розподілом на клієнтську та серверну частини;

– розробити алгоритмічне та інформаційне забезпечення системи;

– реалізувати серверний API для обробки запитів на генерацію та управління базою даних;

– розробити клієнтський вебінтерфейс на базі React.js для взаємодії з системою;

– спроєктувати і реалізувати базу даних MongoDB для зберігання контенту та профілів користувачів;

– розробити модулі генераторів для основних типів HPI-контенту;

– провести тестування функціональності системи та перевірити коректність результатів.

Об'єктом дослідження є процеси отримання, зберігання, обробки, передачі та використання інформації в системах автоматизованого генерування ігрового контенту для HPI.

Предмет дослідження є методи та алгоритми генерування ігрового контенту на основі великих мовних моделей з параметризованим підходом.

Практичне значення результатів полягає у розробці середовища для досвідчених майстрів та новачків в настільно-рольових іграх, щоб оптимізувати процес генерації контенту для ігрових сесій. Розроблена система підтримує набір правил для настільно рольових ігор (D&D 5e, Pathfinder, Blades in the Dark та ін.) та орієнтована на майстрів гри різного рівня досвіду.

Результати роботи опубліковано в матеріалах 4-ї Міжнародної науково-практичної конференції «Scientific Progress: Theories, Applications and Global Impact» (м. Брага, Португалія, 8–10 червня 2026 р.) (додаток Б).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ПРОЕКТУВАННЯ

1.1 Опис предметної області процесів генерації контенту настільних рольових ігор

Настільно-рольові ігри (НРІ, англ. Tabletop Role-Playing Games, TRPG) – жанр ігор, у якому учасники колективно будують ігровий наратив, керуючись системою правил і взаємодіючи зі спільно уявлюваним ігровим світом. Основою ігрового процесу – є рольове відтворення: кожен гравець виступає від імені свого персонажа (PC – Player Character), тоді як майстер гри виступає наратором, суддею та режисером усього, що відбувається поза межами контролю гравців.

Жанр НРІ сформувався в середині 1970-х років із виходом першого видання Dungeons & Dragons (D&D) Гарі Гайгакса та Дейва Арнесона у 1974 році. Від того часу ігрова індустрія значно еволюціонувала: з'явилися десятки різних ігрових систем із різноманітними механіками, антураж та принципами наративу. Сучасні НРІ охоплюють жанровий спектр від класичного фентезі та науково-фантастичних всесвітів до містичних трилерів, кіберпанку та постапокаліпсису.

Ринок НРІ переживає безпрецедентне зростання. За даними дослідницьких агентств ICv2 та Circana [7], обсяг продажів настільно-рольових ігор і супутньої продукції у 2023 році перевищив 1,5 млрд доларів США у всьому світі. Платформа YouTube засвідчила понад 2 мільярди переглядів контенту, пов'язаного з НРІ. Такий ринковий успіх є наслідком культурного феномену Critical Role, а також підвищення загального інтересу до інтерактивних наративних форм розваг.

Ключовою проблемою для майстра гри залишається колосальне навантаження на підготовку ігрових матеріалів. За даними опитування Shea M. (портал Sly Flourish, 3 663 респонденти), близько третини майстрів гри витрачають від 3 годин і більше на підготовку чотиригодинної сесії [8]. на підготовку однієї ігрової сесії тривалістю 3–4 години. Це охоплює такі складові:

– розробку та опис локацій (таверни, підземелля, ліси, міста, руїни тощо) з атмосферними деталями;

– створення NPC (Non-Player Characters) – персонажів з іменами, біографіями, мотиваціями та діалогами;

– побудову сюжетних ліній, квестів і загадок з розгалуженими сценаріями;

– балансування бойових сутичок відповідно до рівня та складу партії гравців;

– написання описів атмосфери для глибшого занурення у світ гри.

З точки зору інформаційних систем процес підготовки HPI-контенту є інформаційним бізнес-процесом із чітко визначеною структурою вхідних та вихідних даних. Вхідними даними є параметри генерації, задані МГ (жанр, тип контенту, система правил, складність тощо). Вихідними – структурований текстовий контент, готовий до використання під час ігрової сесії. У таких умовах доцільним є застосування автоматизованих інформаційних систем, здатних забезпечити структурований підхід до генерування ігрових матеріалів.

Для формалізації описаного процесу та подальшого проєктування системи доцільно побудувати дерево функцій, яке відображає основні функціональні блоки веб-орієнтованої ІС, що представлено на рисунку 1.1.

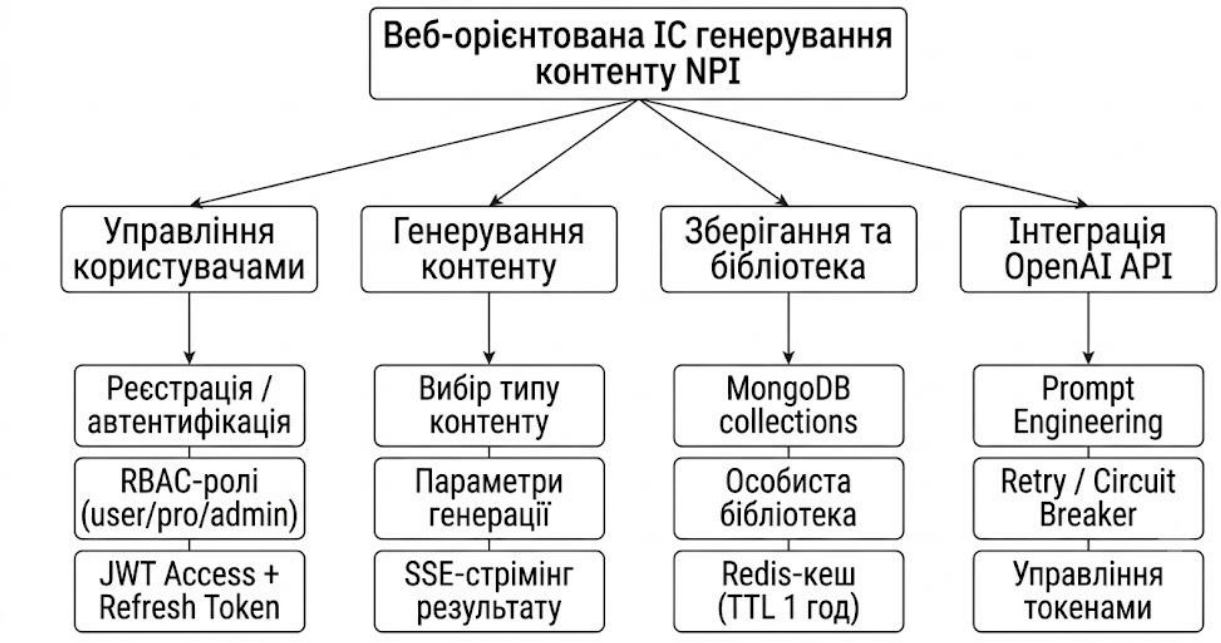


Рисунок 1.1 – Дерево функцій веб-орієнтованої ІС генерування контенту HPI

Наведена структура дерева функцій відображає чотири ключові функціональні блоки системи: управління користувачами (реєстрація, RBAC-ролі, JWT-автентифікація), генерування контенту (вибір типу, параметризація, SSE-стрімінг), зберігання та бібліотека (MongoDB-таблиці, особиста бібліотека, Redis-кеш) та інтеграція з OpenAI API (Prompt Engineering, механізми надійності, управління токенами).

Поряд із традиційним підходом до підготовки матеріалів розвивається концепція процедурної генерації (Procedural Content Generation, PCG) [5] – автоматизованого створення ігрового контенту за допомогою алгоритмів. PCG широко використовується у відеоіграх (Minecraft, No Man's Sky, Dwarf Fortress), однак у контексті настільних ігор залишається відносно нерозвиненим. Поєднання класичних методів PCG із сучасними LLM відкриває нові горизонти можливостей для генерування різноманітного та якісного ігрового контенту.

Сучасні настільно-рольові ігри поділяються на кілька ключових класів за своєю механічною основою, що безпосередньо визначає структуру і обсяг необхідного ігрового контенту.

Клас-орієнтовані системи (D&D 5e, Pathfinder 2e) базуються на чіткій ієрархії класів персонажів, рівнях досвіду та кількісних характеристиках (сила, спритність, інтелект тощо). Для таких систем майстер гри зобов'язаний готувати детально збалансований контент: NPC з прописаними статблоками, підземелля з розрахованими encounter-балансами, артефакти з точними механічними ефектами. Обсяг текстових матеріалів на одну сесію для D&D 5e становить орієнтовно 5–15 сторінок описів.

Наративні системи (Blades in the Dark, Apocalypse World, Ironsworn) ставлять у пріоритет не числові характеристики, а якість наративу, мотивацій персонажів та атмосферу. Тут МГ потребує насамперед яскравих описів локацій, емоційно переконливих NPC та сюжетних зачіпок – але без складних статблоків. Генерований контент має бути стилістично витончений і лаконічний.

Безкласові та безрівневі системи (GURPS, Savage Worlds, Call of Cthulhu) передбачають більшу свободу у конструюванні персонажів та світів, але водночас

вимагають від МГ гнучкого підходу до опису NPC та навколишнього середовища без прив'язки до стандартних шаблонів.

Така різноманітність ігрових систем ставить вимогу до системи генерування контенту: підтримувати параметризований підхід, при якому МГ може обирати конкретну систему правил, і на основі цього вибору система адаптує структуру вихідного контенту.

Роль майстра гри є унікальною в ігровому дизайні. На відміну від відеоігор, де контент генерується наперед розробниками, у HPI МГ виконує функцію творчого продюсера в реальному часі: він реагує на дії гравців, імпровізує, підтримує наратив та забезпечує внутрішньоігрову логіку.

З функціональної точки зору весь ігровий контент, що готується МГ, можна класифікувати за такими категоріями:

– просторовий контент: описи локацій (зовнішній вигляд, атмосфера, ключові орієнтири, приховані деталі): таверни, замки, підземелля, ліси, міста;

– соціальний контент: NPC з іменами, зовнішністю, рисами характеру, мотиваціями та типовими репліками. Якісний NPC є одним із ключових факторів занурення гравців у світ;

– сюжетний контент: квести, зав'язки сюжетних ліній, загадки, таємниці, вузлові події;

– механічний контент: статблоки монстрів, характеристики предметів, артефакти з магічними ефектами;

– атмосферний контент: флейвор-тексти, описи звуків та запахів, монологи злодіїв, легенди та перекази.

Кожна з цих категорій вимагає специфічного підходу при автоматизованому генеруванні. Наприклад, генерація NPC має враховувати жанр гри, рівень складності кампанії та поточний контекст сесії. Генерація підземелля – архітектурну логіку, розміщення пасток та монстрів відповідно до рівня партії. Це підтверджує необхідність типологізованої системи генераторів із відповідними параметрами для кожного типу контенту.

Для глибшого розуміння потреб цільової аудиторії та формалізації вимог до майбутньої системи доцільно розглянути типові сценарії роботи майстра гри з

ігровими матеріалами – від підготовки до сесії до імпровізації в її процесі.Сценарії взаємодії майстра гри з ігровим контентом:

– сценарій 1: планова підготовка до кампанії: досвідчений МГ планує тижневу сесію у жанрі high fantasy (D&D 5e, рівень партії 7), йому необхідно підготувати трьох NPC-торговців, опис таверни, підземелля на 4–5 кімнат та артефакт-нагороду, за ручного підходу підготовка займає від 4 до 6 годин, більшість яких витрачається не на творчі рішення, а на рутинне оформлення статблоків, описів, імен та шаблонів;

– сценарій 2: початківець-МГ, перша кампанія: новачок стикається з труднощами вже на етапі підготовки, оскільки відсутність досвіду не дозволяє швидко генерувати якісні описи локацій чи цікаві мотивації для NPC, брак інструментів змушує або спрощувати гру до примітивних сценаріїв, або витрачати надмірно багато часу, що є одним із ключових факторів, які відлякують потенційних МГ від ведення ігор;

– сценарій 3: імпровізація під час сесії: під час активної гри гравці несподівано звертаються до незапланованого персонажа або вирішують дослідити локацію, якої не існувало в плані, МГ змушений імпровізувати в реальному часі, що потребує значної творчої концентрації й нерідко призводить до поверхневих та непослідовних описів.

Аналіз цих сценаріїв дозволяє виокремити два ключових режими роботи з ігровим контентом: режим підготовки (асинхронне генерування та зберігання матеріалів наперед) та оперативний режим (швидке генерування під час сесії з мінімальною кількістю вхідних параметрів). Ефективна інформаційна система має підтримувати обидва режими, забезпечуючи при цьому зручність інтерфейсу, швидкодію та стилістичну якість результатів.

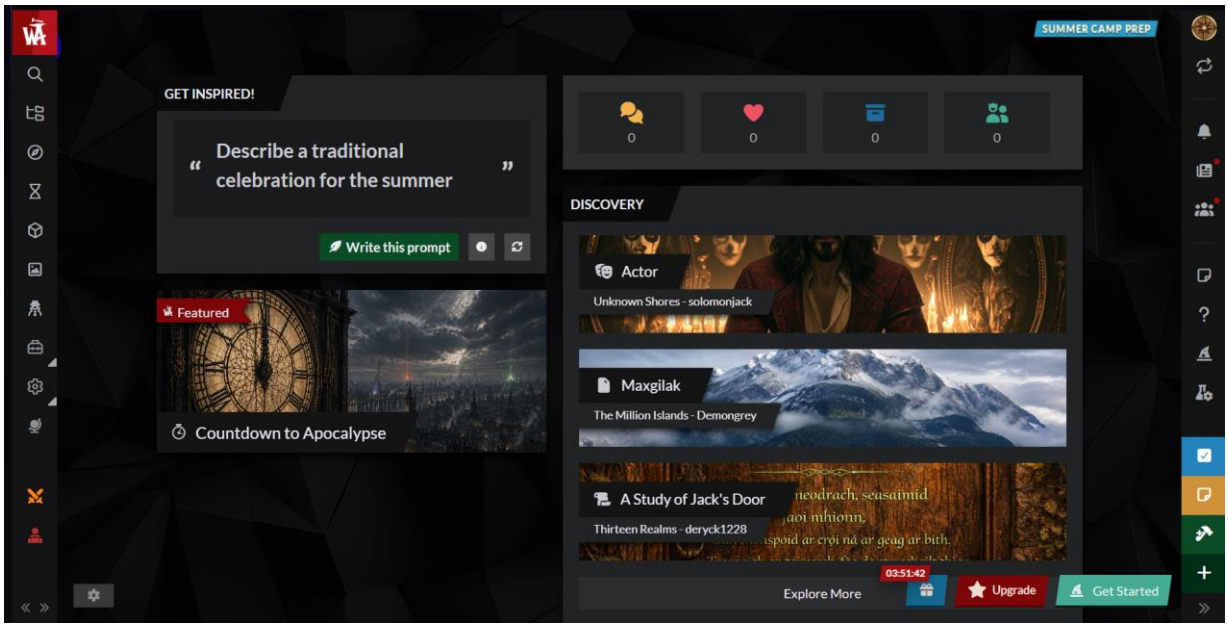
Таким чином, предметна область HPI є достатньо сформованою, має чітку структуру потреб цільової аудиторії та об'єктивно потребує сучасних технологічних рішень для автоматизації рутинних завдань майстра гри.

1.2 Огляд і аналіз існуючих рішень

На сучасному ринку програмного забезпечення для настільно-рольових ігор існує ряд інструментів, що частково вирішують задачу генерування контенту. Аналіз наявних рішень дає змогу оцінити їхні функціональні можливості, виявити переваги та обмеження, а також сформулювати вимоги до проєктованої системи.

World Anvil (worldanvil.com) [2] – платформа для побудови ігрових світів (worldbuilding), яка надає МГ засоби для структурування та зберігання елементів ігрового всесвіту: персонажів, локацій, організацій, хронологій та артефактів. Платформа пропонує зручний інтерфейс з шаблонами та інструменти для генерування карт. Однак World Anvil не має функції автоматичного генерування текстового контенту – усі описи пишуться МГ вручну. Платформа орієнтована на структурування вже наявних матеріалів, а не на їхнє автоматизоване створення.

На рисунку 1.2 показано інтерфейс Warld Anvil стилізований під фантастичну тематику. У даному інтерфейсі платформи можна побачити головну сторінку створення свтіу користувача, інструменти для пошуку інформації та статистика стосовно відвідувачів вигаданого світу користувача.



Риснок 1.2 – Інтерфейс Warld Anvil

Donjon (donjon.bin.sh) [3] – класичний безкоштовний генератор контенту для HPI. Інструмент пропонує генератори підземель, карт міст, імен NPC, лутових таблиць та базових описів персонажів. Генерація відбувається за шаблонами та таблицями випадкових значень – без застосування штучного інтелекту. Результати є механічними, позбавленими стилістичної цілісності та контексту.

На рисунку 1.3 показано інтерфейс Donjon з простим оформленням. В Dongon немає особистого кабінету для зберігання та редакцій сгенерованого контенту. Платформа пропонує нам купу інструментів для генерації інформації у малих обсягах стосовно конкретних речей.

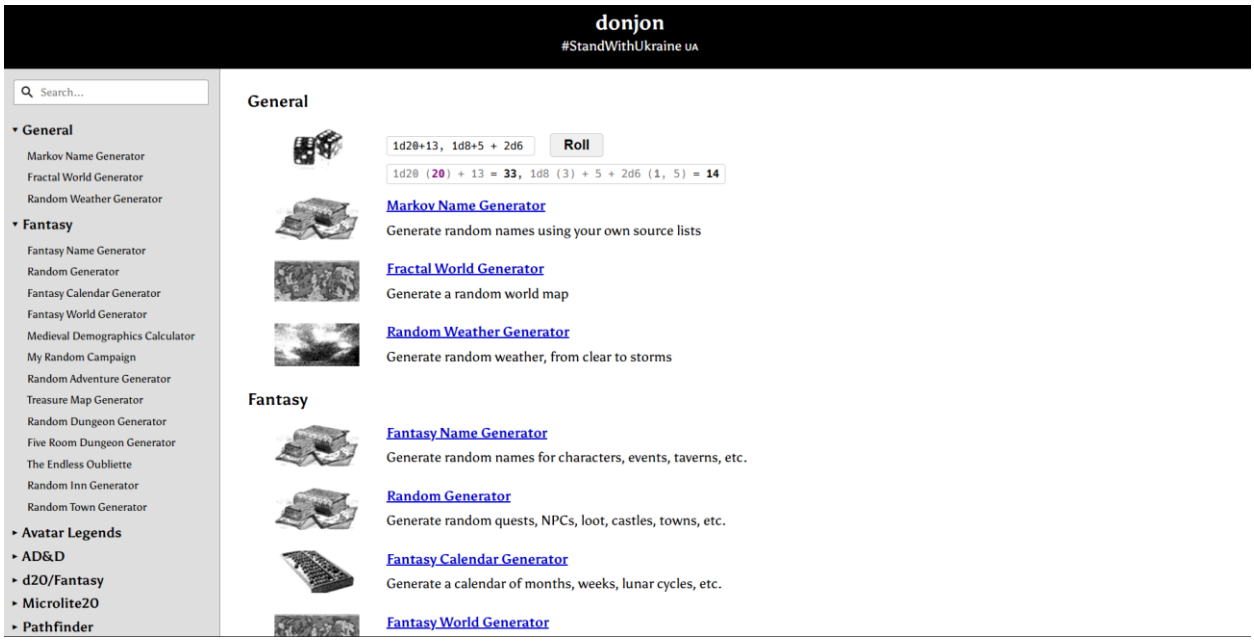


Рисунок 1.3 – інтерфейс Donjon

Perchance (perchance.org) [9] – платформа для побудови генераторів на основі власної мови розмітки. Технічний підхід є потужним, але вимагає від МГ значних технічних знань для налаштування генераторів. Наявний ШІ-функціонал обмежений і слабо інтегрований в основний функціонал платформи.

На рисунку 1.4 показано інтерфейс Perchance.

Хмарні сервіси типу Masterpiece Generator використовують GPT-подібні моделі для генерування описів персонажів та локацій. Ці інструменти пропонують спрощений інтерфейс і обмежені можливості налаштування. Відсутня інтеграція з

системами правил конкретних HPI, брак персистентності та неможливість редагування і повторного використання – основні недоліки цієї категорії рішень.

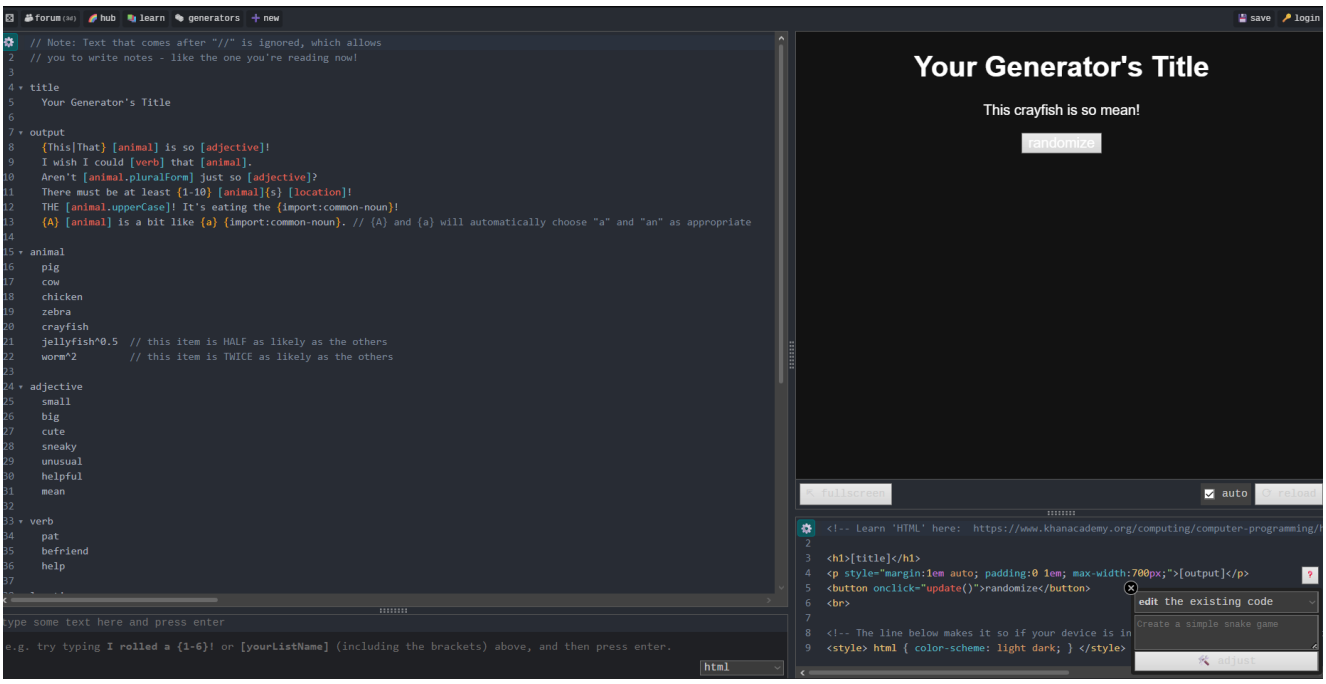


Рисунок 1.4 – Інтерфейс Perchance

Порівняльний аналіз розглянутих рішень наведено у таблиці 1.1.

Окрему категорію складають інструменти, що намагаються інтегрувати штучний інтелект безпосередньо в процес підготовки до гри. Варто розглянути найбільш показові приклади.

Таблиця 1.1 – Порівняльний аналіз існуючих рішень для генерування контенту HPI

Критерій	World Anvil	Donjon	Perchance	Пропонована система
ШІ-генерація тексту	Ні	Ні	Частково	Так (GPT-4)
Збереження контенту	Так	Ні	Ні	Так (MongoDB)

Продовження таблиці 1.1

Параметризація	Обмежена	Часткова	Так	Розширена
Стилістична якість	–	Низька	Середня	Висока
REST API	Так	Ні	Ні	Так (/api/v1)
Безкоштовний тариф	Обмежений	Так	Так	Так

ChatGPT / Claude (загальні чат-боти) активно використовуються МГ як неспеціалізовані інструменти генерації. Значна частина майстрів самостійно формулює запити до GPT-4 або Claude для отримання описів NPC, локацій чи квестів. Однак відсутність спеціалізованих шаблонів, неможливість збереження результатів, відсутність параметричного введення та необхідність ручного форматування суттєво знижують ефективність цього підходу. МГ витрачає додатковий час на «навчання» моделі контексту кожного нового запиту.

Окрему категорію складають інструменти, що намагаються інтегрувати штучний інтелект безпосередньо в процес підготовки до гри. Варто розглянути найбільш показові приклади.

Aidungeon (aidungeon.com) – платформа інтерактивного оповідання на базі LLM, де ШІ виступає спільним автором наративу. Хоча платформа орієнтована на розвагу, а не на підготовку HPI-матеріалів, певна частина користувачів застосовує її для генерації заготовок. Ключовими обмеженнями є низька контрольованість виводу, відсутність структурованих параметрів та неможливість адаптації під конкретні HPI-системи.

Kobold AI / LM Studio (локальні LLM) – рішення для запуску мовних моделей локально. Ентузіасти збирають власні ланцюжки промптів для генерації HPI-контенту. Цей підхід потребує значної технічної компетентності, налаштування апаратного забезпечення та не підходить для широкої аудиторії МГ.

Таким чином, попри доступність загальних LLM-інструментів, жоден із них

не надає спеціалізованого рішення, адаптованого під специфіку HPI: з типологізованими генераторами, параметричним введенням, підтримкою конкретних ігрових систем та персистентним зберіганням результатів.

Аналіз розглянутих програмних рішень свідчить про те, що жодне з них не поєднує повноцінну ШІ-генерацію якісного тексту, персистентність даних та гнучку параметризацію в єдиній системі. Це підтверджує доцільність розроблення нового комплексного веб-рішення, яке усуває виявлені недоліки.

1.3 Постановка задачі проектування

На основі проведеного аналізу предметної галузі та огляду існуючих рішень сформульовано основну задачу кваліфікаційної роботи: розробка веб-орієнтованої інформаційної системи генерування контенту настільно-рольових ігор із застосуванням технологій штучного інтелекту (OpenAI GPT-4) та сучасного технологічного стеку (React.js, Node.js, MongoDB).

Для досягнення поставленої мети необхідно вирішити такі завдання:

– розробити зручний веб-інтерфейс для МГ, що дозволяє задавати параметри генерації та переглядати, редагувати і зберігати контент;

– реалізувати серверний API для обробки запитів, формування промптів для GPT-4 та управління базою даних;

– спроєктувати базу даних MongoDB для зберігання шаблонів контенту, збережених генерацій та профілів;

– розробити систему генераторів для основних типів HPI-контенту: NPC, локацій, квестів, таверн, підземель та артефактів;

– реалізувати автентифікацію користувачів та персональний кабінет із бібліотекою збереженого контенту.

Функціональні вимоги до системи охоплюють: генерацію шести типів ігрового контенту з розширеною параметризацією, збереження результатів у персональній бібліотеці, JWT-автентифікацію [17], підтримку кількох HPI-систем (D&D 5e, Pathfinder, Blades in the Dark та ін.), SSE-стрімінг результатів у реальному часі та систему шаблонів промптів.

Нефункціональні вимоги:

– час відповіді API не більше 2 секунд для стандартних запитів; час генерації контенту не більше 15 секунд;

– стабільність при 100 одночасних користувачах;

– безпечне зберігання паролів (bcrypt);

– захист від XSS та ін'єкційних атак.

2 АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ ВЕБ-ОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ГЕНЕРУВАННЯ КОНТЕНТУ НАСТІЛЬНО-РОЛЬОВИХ ІГОР

2.1 Загальна структура веб-орієнтованої ІС

З огляду на поставлену задачу та визначені вимоги до системи генерування контенту настільно-рольових ігор у межах її реалізації обрано клієнт-серверний підхід. Загальна архітектура системи побудована за принципом Client-Server з чітким розподілом відповідальностей між рівнями: клієнтським (React.js SPA), рівнем API-шлюзу (Node.js/Express), рівнем бізнес-логіки (сервіси генерування), рівнем даних (MongoDB + Redis) та рівнем зовнішньої інтеграції (OpenAI API). Такий підхід відповідає принципам Clean Architecture та Separation of Concerns [6], що забезпечує незалежність кожного шару та можливість їх окремого тестування і заміни.

Загальна концептуальна схема п'ятирівневої клієнт-серверної архітектури представлена на рисунку 2.1.

Рисунок 2.1 – Клієнт-серверна архітектура веб-орієнтованої ІС

Клієнтська частина (рівень 1) виступає зовнішньою оболонкою системи, відповідає за взаємодію з користувачем, збір параметрів генерування і відображення результатів. Реалізована вона як Single Page Application (SPA) на базі React.js версії 18. Маршрутизація здійснюється засобами React Router v6 з підходом lazy loading для оптимізації часу завантаження. Управління серверним станом реалізовано через TanStack Query (React Query v5)[31], що забезпечує кешування, автоматичне повторне виконання запитів при помилках та оптимістичні оновлення.

Рівень API-шлюзу (рівень 2) реалізовано на Node.js 20 LTS із використанням Express.js. Цей рівень приймає HTTP-запити, виконує JWT-автентифікацію, валідацію вхідних даних та маршрутизацію до відповідних контролерів. Middleware-конвеєр включає CORS, Helmet.js, Morgan та Rate Limiter (100 запитів за 15 хвилин).

Рівень бізнес-логіки (рівень 3) містить сервіси, що реалізують основну логіку застосунку: GenerationService (формування промптів та взаємодія з OpenAI), UserService (управління користувачами), LibraryService (операції з бібліотекою) та TemplateService (управління шаблонами).

Рівень даних (рівень 4) забезпечується MongoDB Atlas із Mongoose ODM. Документо-орієнтована модель природно відображає різнорідну структуру ігрового контенту. Для кешування результатів генерувань використовується Redis із TTL [22] 1 година.

Рівень зовнішньої інтеграції (рівень 5) забезпечує взаємодію з OpenAI API через офіційну бібліотеку openai для Node.js. Для надійності реалізовано патерн Retry with Exponential Backoff та Circuit Breaker.

Для відображення повної загальної структури та логіки взаємодії компонентів використано модель у стандарті IDEF0 (рисунок 2.2), яка демонструє, як усі рівні архітектури взаємодіють для обробки запитів користувачів, виконання генерування контенту та повернення результатів.

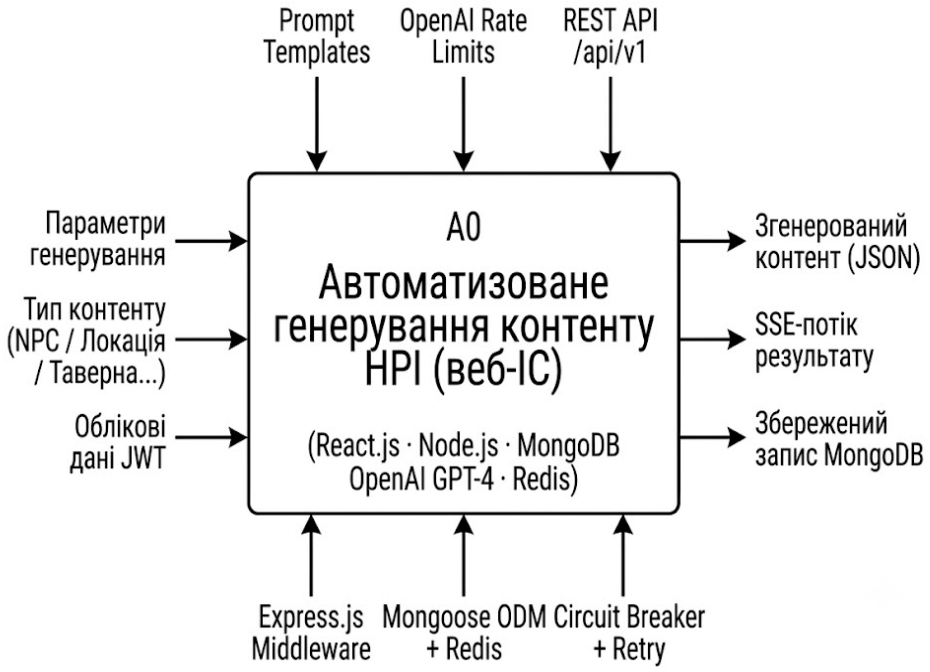


Рисунок 2.2 – Функціональна модель веб-орієнтованої ІС (IDEF0-діаграма, рівень A0)

Відповідно до поданої IDEF0-діаграми, центральним процесом є «Автоматизоване генерування контенту HPI (веб-ІС)». Вхідними даними виступають параметри генерування від МГ, тип контенту та облікові дані JWT. Управління процесом визначається шаблонами промптів, обмеженнями OpenAI API та версіонованим REST API (/api/v1) [16]. Механізмами є Express.js middleware, Mongoose ODM + Redis та патерни надійності Circuit Breaker + Retry. Вихідними даними є структурований JSON-контент, SSE-потік результату та збережений запис у MongoDB.

Структура клієнтської частини організована відповідно до методології Feature-Sliced Design (FSD) [36]: app/ (конфігурація), pages/ (GeneratorPage, LibraryPage, AuthPage), features/ (ізольовані модулі генераторів), entities/ (Character, Location, Quest, Artifact) та shared/ (спільні утиліти та UI-компоненти). Серверна архітектура побудована за принципом Layered Architecture: Presentation Layer, Business Logic Layer, Data Access Layer та Infrastructure Layer.

2.2 Алгоритмічне забезпечення

Алгоритм розглядається не лише як послідовність дій, а як ядро логіки функціонування сучасної інтелектуальної системи. У розроблюваній веб-орієнтованій ІС алгоритмічне забезпечення визначає процес обробки запиту МГ та формування якісного ігрового контенту за допомогою великих мовних моделей.

Центральним компонентом системи є модуль генерування, що відповідає за перетворення параметрів від МГ на якісний ігровий текст. Алгоритм генерування складається з таких етапів:

– валідація вхідних параметрів через Joi-схему;

– збагачення контексту – додавання знань про обране середовище та антураж або систему правил;

– формування промпту (PromptBuilder);

– виклик OpenAI GPT-4 API;

– парсинг та валідація JSON-відповіді;

– збереження в MongoDB та повернення клієнту.

Схема алгоритму генерування ігрового контенту наведена на рисунку 2.3.

Відповідно до поданої схеми алгоритм розпочинається з отримання запиту від МГ. Система перевіряє JWT-токен (у разі відхилення повертається відповідь 401) та виконує валідацію параметрів за Joi-схемою. Далі здійснюється перевірка Redis-кешу: за наявності cache hit результат повертається негайно без звернення до OpenAI. Після формування промпту та виклику GPT-4 API виконується перевірка успішності відповіді: у разі збою активуються патерни Retry та Circuit Breaker. Успішна відповідь парситься, зберігається в MongoDB та повертається клієнту через SSE-стрімінг.

Окремої уваги заслуговує архітектура модуля PromptBuilder –компонента, відповідального за формування структурованих запитів до OpenAI GPT-4. Якість та передбачуваність генерованого контенту безпосередньо залежить від точності та повноти промпту, тому цей компонент є критично важливим з точки зору алгоритмічного забезпечення.

Рисунок 2.3 – Схема алгоритму генерування ігрового контенту

Кожен промпт у системі будується за триступеневою схемою:

– системний промпт (System Prompt) – встановлює роль моделі («Ти є досвідченим майстром гри та письменником у жанрі фентезі…»), визначає

обов'язковий формат відповіді (JSON-схема з переліком полів), мову виводу та стилістичні обмеження. Системний промпт є фіксованим для кожного типу контенту і не змінюється залежно від параметрів користувача;

– контекстний блок (Context Injection) – динамічно додає знання про обрану HPI-систему (механіки D&D 5e, Pathfinder тощо), жанрові особливості антуражу та поточний рівень складності. Цей блок збагачує запит специфічними для системи правилами та термінологією;

– параметричний блок (User Prompt) – містить конкретні значення, введені МГ: тип контенту, назву, ключові риси, антураж, складність, спеціальні побажання.

Таке трирівневе розмежування забезпечує дві ключові властивості системи: стабільність структури відповіді (оскільки формат завжди визначено на рівні системного промпту) та гнучкість контенту (оскільки параметричний блок повністю визначається введенням МГ). Шаблони зберігаються у колекції templates MongoDB, що дозволяє адміністраторам оновлювати їх без змін у коді.

Ключовим аспектом алгоритмічного забезпечення є розробка промптів. Система шаблонів промптів (Prompt Templates) базується на поєднанні системного промпту (System Prompt), що задає роль та обов'язковий JSON-формат відповіді, та користувацького промпту (User Prompt), що передає конкретні параметри. Таке розмежування забезпечує стабільність структури відповідей та їх програмну оброблюваність. Гіперпараметри запитів до GPT-4 підбирались емпірично. Параметр temperature для генерування NPC оптимальним є значення 0.85 (унікальні особистості без втрати структурованості); для бойових статистик – 0.4 (числова коректність). Параметр max_tokens: 1500 для NPC, 2000 для локацій та підземель, 800 для артефактів.

Для забезпечення надійності реалізовано два патерни стійкості. Retry with Exponential Backoff: перша спроба – через 1–2 с, друга – 2–4 с, третя – 4–8 с (з random jitter). Circuit Breaker: після 5 послідовних збоїв протягом 60 с блокує запити на 30 с, потім переходить у напіввідкритий стан для тестового запиту. Алгоритм автентифікації реалізовано за схемою Access Token (15 хв.) + Refresh Token (30 днів) із RBAC-авторизацією.

2.3 Інформаційне забезпечення

Інформаційне забезпечення є фундаментальною складовою будь-якої інформаційної системи і визначає структуру, організацію та спосіб зберігання даних, необхідних для безперебійного функціонування всіх компонентів. Воно включає різні типи даних – структуровані, напівструктуровані та неструктуровані – механізми їх обробки, зберігання і захисту, а також опис вхідних і вихідних даних.

У межах розроблюваної веб-орієнтованої ІС кожен тип даних відповідає за окремий функціональний сегмент. Структуровані дані зберігаються в MongoDB у вигляді документів із чітко визначеними полями: профілі користувачів, їхні ролі та підписки. Напівструктуровані дані представлені JSON-відповідями OpenAI API та конфігураційними файлами застосунку. Неструктуровані дані включають довільні текстові описи антуражу від МГ та згенерований контент.

MongoDB [11] обрано як основну СУБД з таких міркувань. По-перше, документо-орієнтована модель природно відображає різнорідну структуру ігрового контенту: опис NPC є документом із вкладеними об'єктами, тоді як опис підземелля – документ із масивом вкладених документів-кімнат. По-друге, відсутність жорсткої схеми (schema flexibility) дозволяє різним типам контенту мати різний набір полів. По-третє, MongoDB Atlas надає безкоштовний tier для розробки та тестування.

ER-модель бази даних наведена на рисунку 2.4.

ER-модель бази даних формують таблиці, що детально описані нижче.

Таблиця users зберігає профілі користувачів: _id, email (unique), passwordHash (bcrypt, cost factor 12), username, role (user|pro|admin), tokensRemaining та createdAt. Ця таблиця є точкою входу, до якої прив'язаний весь індивідуальний досвід користувача.

Таблиця generations є центральним сховищем усього згенерованого контенту. Використання дискримінованих типів (contentType: enum npc|location|tavern|dungeon|quest|artifact) дозволяє зберігати різні типи в одній Таблиця з полями: userId (FK→users), params (Mixed), content (Mixed), isFavorite,

tags, tokensUsed. Складений індекс { userId: 1, contentType: 1, createdAt: -1 } забезпечує швидку пагінацію бібліотеки.

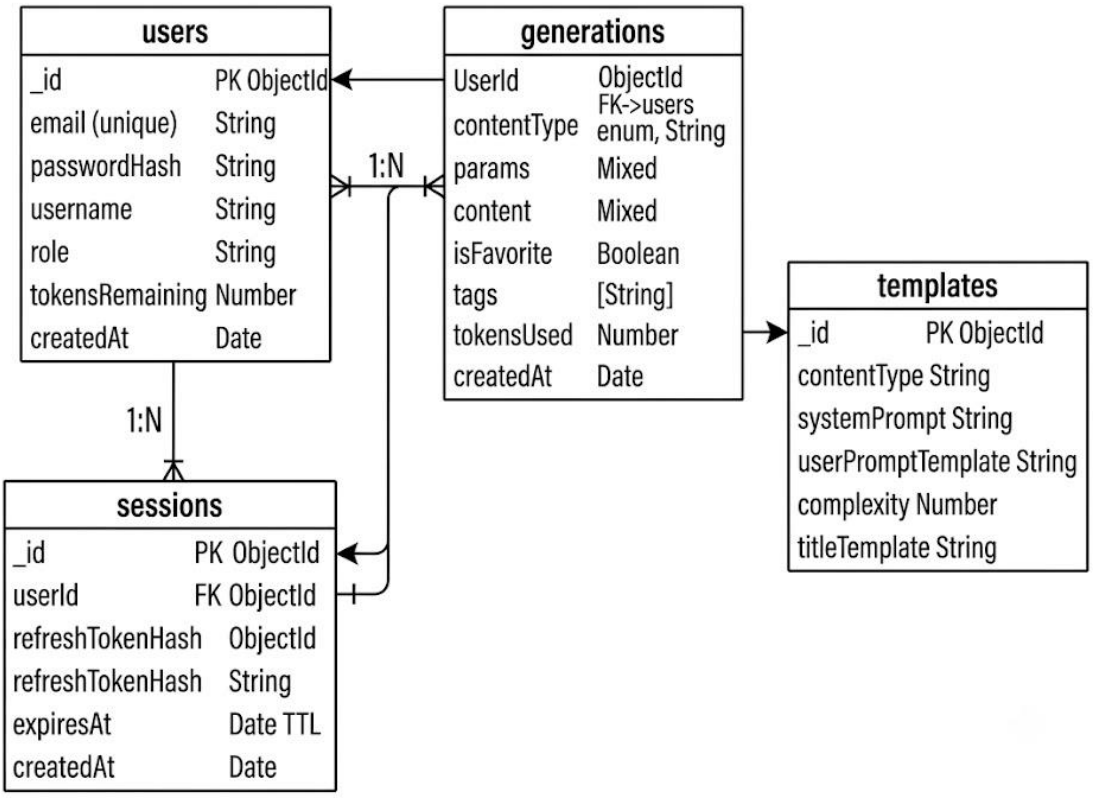


Рисунок 2.4 – ER-модель бази даних веб-орієнтованої ІС

Таблиця templates зберігає шаблони промптів: системний промпт (systemPrompt), шаблон користувацького промпту, параметри за замовчуванням, рейтинг та createdBy (FK–users). Передбачено публічні та приватні шаблони.

Таблиця sessions зберігає Refresh Token у вигляді bcrypt-хешу із TTL-індексом на поле expiresAt для автоматичного видалення прострочених записів.

Характеристики основних таблиць MongoDB наведено у таблиці 2.2.

Вхідними даними системи є: параметри генерування від МГ (тип контенту, жанр, антура, складність, система правил, мова), довільний текстовий опис контексту та дані автентифікації. Вихідними даними є структурований JSON-об'єкт згенерованого контенту та відформатований контент для відображення в інтерфейсі.

Таблиця 2.2 – Характеристики таблиць БД

Таблиці БД	Середній розмір документа	Очікувана кількість	Ключові індекси
users	~2 KB	10 000+	email (unique)
generations	~5–15 KB	500 000+	(userId, contentType, createdAt)
templates	~3 KB	1 000+	(contentType, isPublic, rating)
sessions	~1 KB	50 000+	expiresAt (TTL)

Для забезпечення прийнятної швидкодії системи при значному навантаженні реалізовано дворівневу стратегію кешування.

Перший рівень – Redis-кеш генерувань. Оскільки генерування контенту через OpenAI API є відносно дорогою операцією (як за часом – до 15 секунд, так і за вартістю токенів), система кешує результати із однаковим набором параметрів. Ключ кешу формується як хеш об'єкта параметрів запиту (contentType + params). TTL кешу становить 1 годину. Це дозволяє повертати повторювані запити миттєво без повторного звернення до OpenAI API.

Другий рівень – кешування запитів до MongoDB. TanStack Query на клієнтській стороні кешує результати GET-запитів до бібліотеки та шаблонів із налаштованим staleTime 5 хвилин, що суттєво зменшує кількість запитів до сервера при навігації між сторінками.

Оптимізація запитів до MongoDB забезпечується складеними індексами, описаними в таблиці 2.2. Зокрема, індекс { userId: 1, contentType: 1, createdAt: -1 } на колекції generations забезпечує пагінацію особистої бібліотеки користувача навіть при великій кількості збережених генерувань. TTL-індекс на полі expiresAt колекції sessions автоматично видаляє прострочені refresh-токени без додаткового

планувальника задач.

Таким чином, інформаційне забезпечення системи об'єднує структуровані дані користувачів, напівструктуровані JSON-відповіді API та неструктурований згенерований контент у єдиній MongoDB-базі. Продумана схема з відповідними індексами та Redis-кешем забезпечує продуктивність і масштабованість системи.

3 РЕАЛІЗАЦІЯ ВЕБ-ОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ГЕНЕРУВАННЯ КОНТЕНТУ НАСТІЛЬНО-РОЛЬОВИХ ІГОР

3.1 Реалізація програмного забезпечення

На основі спроєктованих рішень програмна реалізація веб-орієнтованої ІС передбачає створення повноцінної клієнт-серверної системи з п'ятьма взаємопов'язаними рівнями. Клієнтська частина реалізована як Single Page Application (SPA) на базі React.js [18] версії 18. Структуру проєкту організовано відповідно до методології Feature-Sliced Design (FSD): app/ (конфігурація провайдерів та ініціалізація), pages/ (GeneratorPage, LibraryPage, AuthPage, ProfilePage), features/ (npcGenerator, locationGenerator, questGenerator тощо), entities/ (Character, Location, Quest, Artifact) та shared/ (спільні утиліти та UI-компоненти).

Серверна частина реалізована на Node.js версії 20 LTS з Express.js [19]. Middleware-конвеєр налаштовано у такому порядку: CORS middleware [33], helmet.js (безпечні HTTP-заголовки), express.json(), morgan (логування), rateLimiter (100 запитів за 15 хвилин), authenticateToken (JWT), validateRequest (Joi-схеми), errorHandler (глобальний обробник). REST API організовано з версіонуванням /api/v1.

Для наочного відображення статичної структури серверної частини системи та взаємозв'язків між основними програмними сутностями розроблено UML-діаграму класів (рисунок 3.1).

На поданій діаграмі класи розподілені за рівнями відповідно до Layered Architecture. Контролери (AuthController, GenerationController, LibraryController, TemplateController) відповідають за обробку HTTP-запитів та делегування бізнес-логіки сервісному шару. Сервіси (UserService, GenerationService) реалізують основну логіку застосунку. GenerationService взаємодіє з OpenAIClient (реалізує Retry + Circuit Breaker) та GenerationRepository (інкапсулює операції з MongoDB).

Основні групи маршрутів REST API: /api/v1/auth (реєстрація, вхід, оновлення токену), /api/v1/generate (генерування контенту та SSE-стрімінг), /api/v1/library (бібліотека збережених генерувань), /api/v1/templates (шаблони промптів), /api/v1/users (профіль та налаштування).

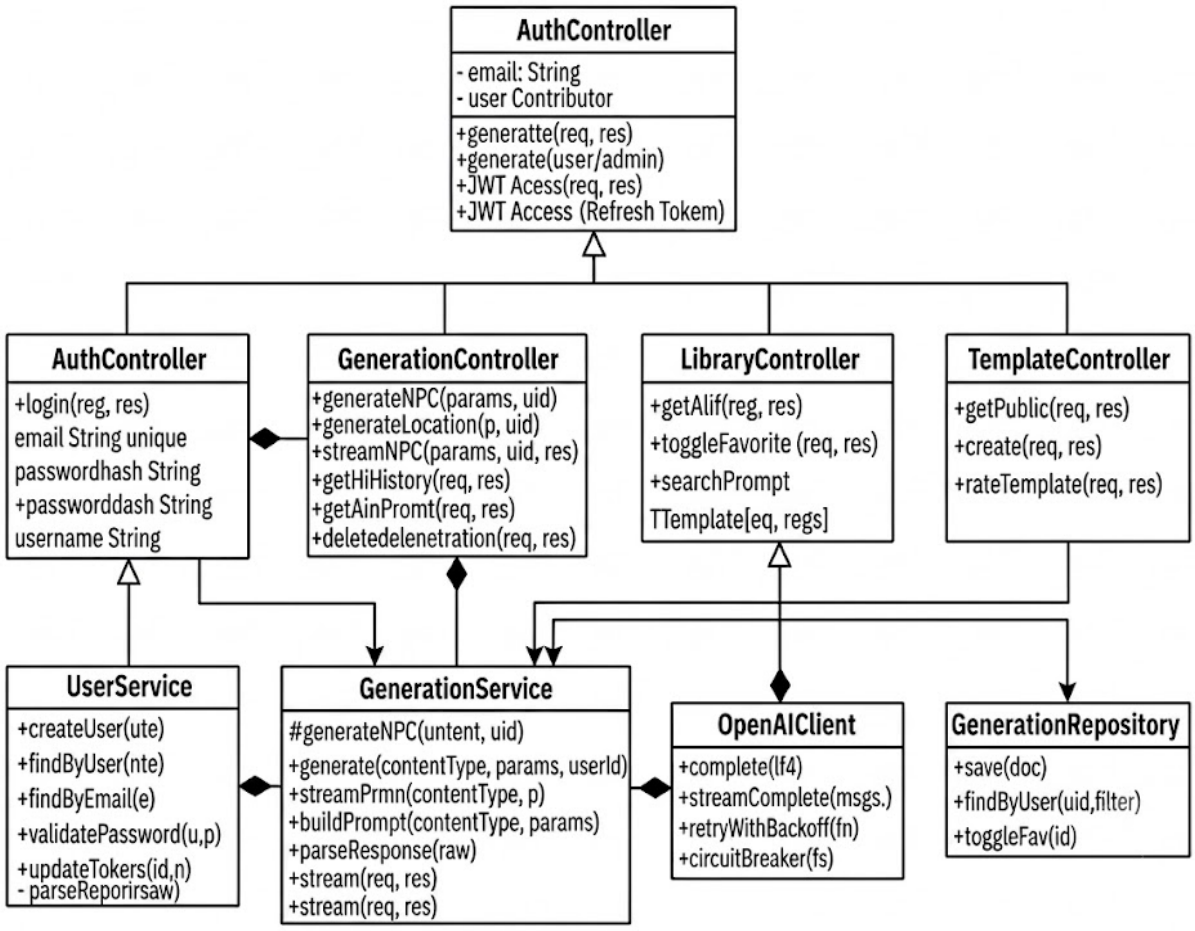


Рисунок 3.1 – UML-діаграма класів серверної частини (контролери та сервіси)

Центральним компонентом є GenerationService, що реалізує формування промптів через PromptBuilder та виклик OpenAI API через OpenAIClient. Клас OpenAIClient реалізує патерни Retry with Exponential Backoff та Circuit Breaker. Для потокової передачі результатів реалізовано /stream endpoint із SSE (Server-Sent Events), що дозволяє відображати текст у реальному часі.

Взаємодія з MongoDB реалізована через Mongoose ODM [20]. Mongoose-схеми визначають структуру документів із валідацією типів, pre/post-хуками та virtual-полями. Підключення реалізовано з пулом з'єднань розміром 10. Розгортання системи здійснено засобами Docker Compose [21]: два мікросервіси (клієнт на Node/Nginx, сервер на Node.js 20-slim) у власних ізольованих контейнерах із спільною внутрішньою мережею.

Ключовим функціональним компонентом серверної частини є система спеціалізованих генераторів – окремих модулів для кожного з шести типів ігрового контенту. Кожен генератор реалізовано як ізольований клас, що наслідує абстрактний базовий клас BaseGenerator та перевизначає метод buildPrompt(). Такий підхід відповідає принципу Open/Closed: додавання нового типу контенту не потребує змін у наявному коді, лише створення нового класу-нащадка.

NpcGenerator формує промпт із зазначенням раси, класу, рівня та ролі персонажа. Вихідний JSON включає поля: name, race, class, appearance, personality, motivation, secret, speechPattern та statBlock (якщо обрано D&D 5e або Pathfinder). Для забезпечення унікальності особистості temperature встановлено на рівні 0.85.

LocationGenerator та TavernGenerator генерують просторові описи з атмосферними деталями, переліком ключових NPC та сюжетними зачіпками. Структура JSON включає: name, description, atmosphere, keyFeatures, inhabitants, plotHooks. Для таверн додатково генеруються menuHighlights та rumors.

DungeonGenerator є найскладнішим з генераторів. Він формує багаторівневий JSON із масивом об'єктів-кімнат (rooms), кожна з яких містить: roomId, name, description, exits, encounters, traps та loot. Кількість кімнат задається параметром МГ (від 3 до 10). Для збереження внутрішньої логіки підземелля у промпті явно вказується вимога до зв'язності маршруту та тематичної єдності.

QuestGenerator генерує структуровані квести з полями: title, hook, objective, stages (масив), rewards, complications та alternativeEndings. Параметр complexity (low / medium / high) визначає кількість етапів та рівень розгалуженості сюжету.

ArtifactGenerator формує описи магічних предметів із механічними ефектами. JSON включає: name, type, appearance, lore, properties (механічні ефекти у форматі обраної HPI-системи), attunement та curseIfAny. Temperature для цього генератора встановлено на рівні 0.6 для збереження механічної коректності.

Всі генератори використовують спільний ResponseParser, що виконує валідацію JSON-відповіді OpenAI за попередньо визначеною JSON Schema для кожного типу контенту. У разі невідповідності схемі автоматично ініціюється повторний запит із уточненим промптом (не більше 2 повторних спроб).

3.2 Інтерфейс користувача

Інтерфейс користувача є засобом взаємодії з програмною системою і не менш важливий, ніж програмна реалізація. Від нього залежить зручність роботи, швидкість сприйняття інформації та ефективність виконання завдань майстром гри.

Інтерфейс веб-орієнтованої ІС спроєктований з урахуванням ключових принципів UX-дизайну та орієнтований на цільову аудиторію –майстрів гри різного рівня досвіду. Проєктування виконувалося засобами Figma, що дозволило детально опрацювати компонентну структуру та колірну гаму до програмної реалізації.

Навігаційна структура включає: головну сторінку (GeneratorPage) з повним функціоналом генерування, сторінку бібліотеки (LibraryPage) для управління збереженим контентом, сторінку профілю (ProfilePage) та сторінки автентифікації (AuthPage).

Розроблено головну сторінку розробленої системи GM Codex. Верхня навігаційна панель забезпечує швидкий перехід між усіма типами генерованого контенту: Герой, Квести, NPC, Артефакти, Локації, Таверни, Підземелля та Бібліотека. Центральна частина сторінки містить картки швидкого доступу до шести генераторів: NPC, локацій, таверн, підземель, квестів та артефактів. Кожна картка відображає дату останнього оновлення, короткий опис параметрів генерації та кнопку запуску. Темна кольорова гама інтерфейсу відповідає тематиці фентезійних HPI та забезпечує комфортну роботу у вечірній час – типовий сценарій використання системи майстром гри. (рисунок 3.2).

Для моделювання поведінки інтерфейсу у часі розроблено діаграму послідовності взаємодії клієнта, сервера та OpenAI API (рисунок 3.3).

На поданній діаграмі послідовності відображено повний цикл генерування контенту:

– МГ через браузер надсилає POST-запит з параметрами та JWT;

– сервер виконує валідацію і перевіряє Redis-кеш (при cache miss – звертається до OpenAI);

– GPT-4 повертає SSE-потік фрагментів; сервер пересилає їх клієнту через

EventSource API;

– текст відображається у реальному часі; після завершення генерування результат зберігається в MongoDB.

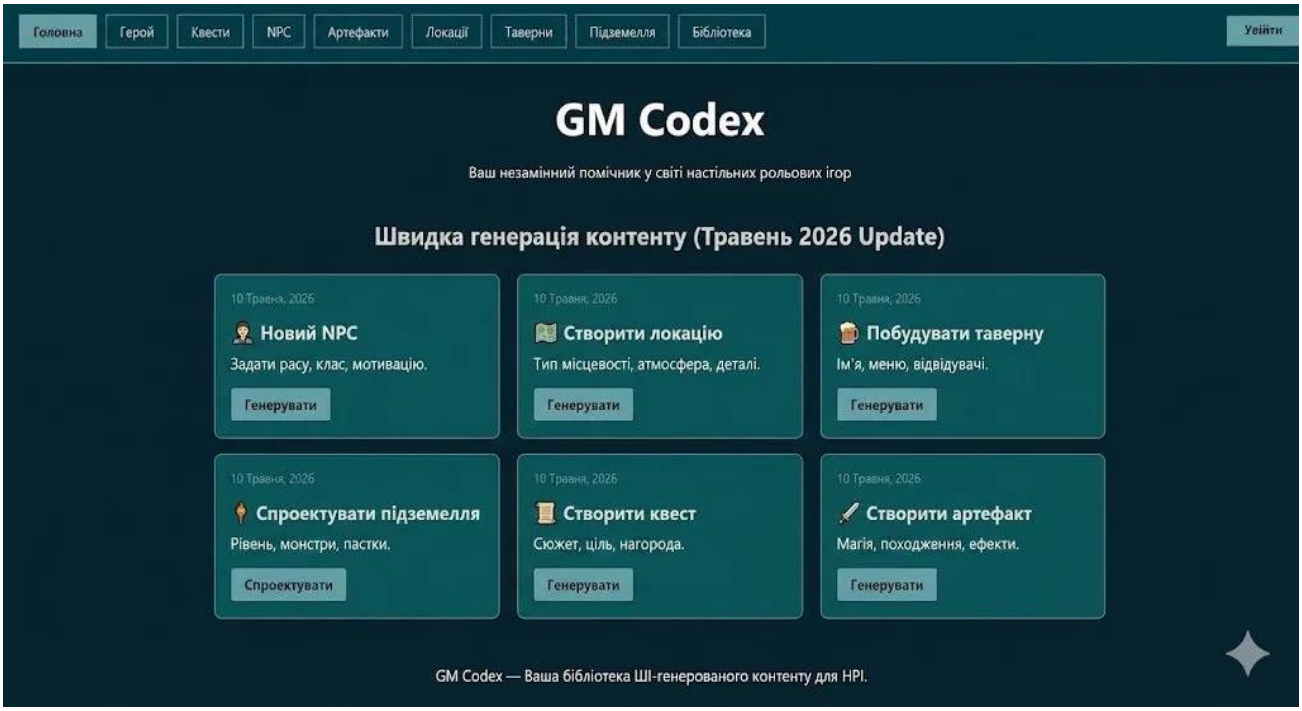


Рисунок 3.2 – Головна сторінка веб-орієнтованої ІС GM Codex

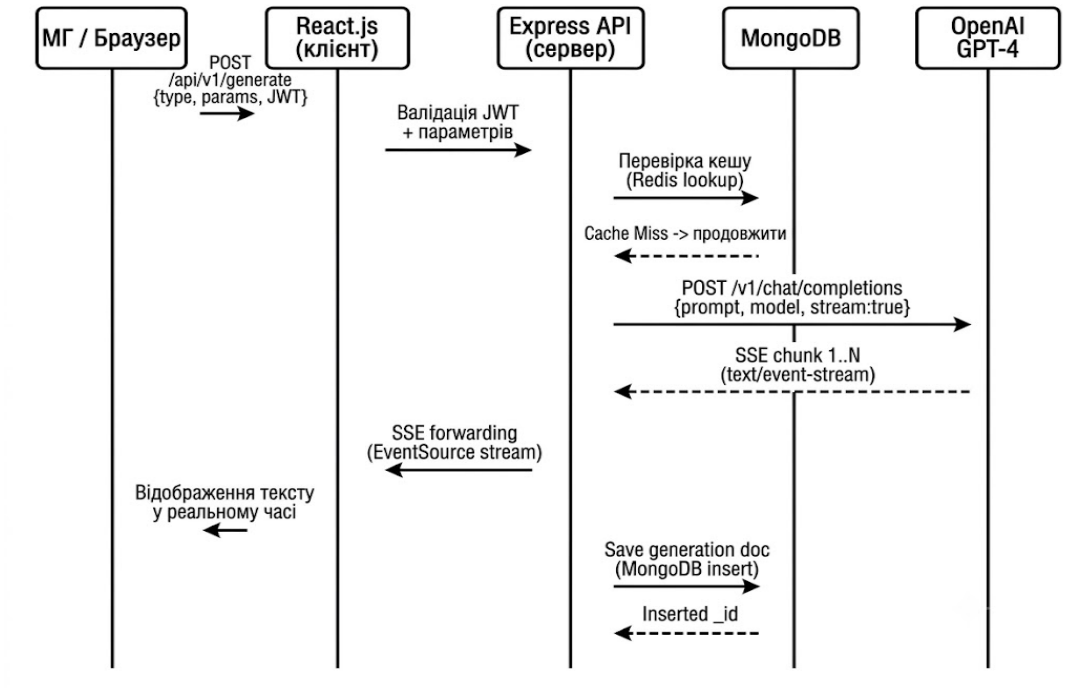


Рисунок 3.3 – Діаграма послідовності генерування контенту (SSE-стрімінг)

Головна сторінка є центральним робочим простором МГ. Вона містить форму вибору типу контенту, панель параметрів (жанр, антураж, система правил, складність, тон, мова), текстове поле для контексту та кнопку запуску генерування. Усі параметри мають значення за замовчуванням (D&D 5e, фентезі, середня складність), що дозволяє почати генерування одним кліком.

Сторінка бібліотеки забезпечує доступ до збереженого контенту з фільтрацією за типом, пошуком за назвою і тегами, сортуванням та пагінацією. Адаптивна верстка на основі Tailwind CSS забезпечує коректне відображення на пристроях різних розмірів.

3.3 Тестування програмного забезпечення

З метою оцінки стабільності та коректності функціонування розробленої веб-орієнтованої ІС здійснено комплексне тестування її основних компонентів. Тестування охоплює модульний, інтеграційний та функціональний рівні відповідно до загальноприйнятої методології верифікації програмного забезпечення.

Модульне тестування серверної частини здійснювалося з використанням Jest та Supertest [35]. Для кожного сервісу розроблено набір юніт-тестів із підстановкою залежностей, що дозволяє тестувати бізнес-логіку ізольовано. Функціональне тестування виконувалося шляхом набору сценаріїв, що охоплюють основні варіанти використання системи. Результати наведено у таблиці 3.1.

Таблиця 3.1 – Результати функціонального тестування

Тестовий сценарій	Очікуваний результат	Фактичний результат	Час
Генерація NPC (D&D 5e, людина, маг)	JSON з полями name, race, appearance тощо	JSON коректний, усі поля присутні	Генерація NPC (D&D 5e, людина, маг)

Продовження таблиці 3.1

Тестовий сценарій	Очікуваний результат	Фактичний результат	Час
Генерація таверни (похмура атмосфера)	Опис з назвою, NPC, гачками	Усі секції присутні, стиль відповідає	<15 с
Збереження в бібліотеку	Запис у MongoDB, відображення	Контент збережено, доступний після перезав.	<1 с
SSE-стрімінг результату	Текст відображається покроково	Потоковий вивід підтверджено	Real-time
Некоректні параметри	Відповідь 422 з описом помилки	422 + перелік помилок валідації	<0.1 с
Реєстрація та вхід	JWT-токени, перехід на головну	Автентифікація успішна	<0.5 с

Результати модульного тестування ключових компонентів наведено у таблиці 3.2.

Таблиця 3.2 – Результати модульного тестування основних компонентів

Модуль / Компонент	Кількість тест	Пройдено	Не пройдено
GenerationService	18	18	0
UserService	12	12	0
PromptBuilder	24	24	0
AuthMiddleware	9	9	0

Продовження таблиці 3.2

Модуль / Компонент	Кількість тест	Пройдено	Не пройдено
UserService	12	12	0
PromptBuilder	24	24	0
AuthMiddleware	9	9	0
Mongoose-схеми	15	15	0
OpenAIClient (mock)	11	11	0

Інтеграційне тестування API здійснювалося за допомогою Supertest та тестової MongoDB-бази (in-memory через mongodb-memory-server). Перевірялася коректність HTTP-відповідей для всіх ендпоінтів, правильність заголовків, валідність JWT-токенів та обробка помилкових запитів.

Навантажувальне тестування проводилося за допомогою Artillery.io [34]. Тест симулював 50 одночасних користувачів протягом 5 хвилин. Середній час відповіді API (без OpenAI) – 87 мс; p95 – 210 мс; p99 – 450 мс. Помилок рівня сервера (5xx) незафіксовано.

Окрему увагу приділено тестуванню якості згенерованого контенту – аспекту, що не охоплюється стандартними методами автоматичного тестування. Оскільки оцінка стилістичної якості та смислової зв'язності тексту є суб'єктивною, розроблено методику ручного оцінювання із залученням цільової аудиторії.

Для оцінювання було сформовано вибірку з 30 генерацій – по 5 для кожного з шести типів контенту – із використанням різних параметрів (жанрів, HPI-систем, рівнів складності). Оцінювання проводилося за чотирма критеріями за шкалою від 1 до 5: структурна повнота (наявність усіх обов'язкових полів), стилістична якість (відповідність жанру та атмосфері), практична придатність (можливість використати результат без суттєвої правки) та внутрішня зв'язність (логічна узгодженість частин контенту).

Результати ручного оцінювання якості контенту наведено у таблиці 3.3.

Таблиця 3.3 – Результати оцінювання якості згенерованого контенту

Тип контенту	Структурна повнота	Стилістична якість	Практична придатність	Внутрішня зв'язність
NPC	5.0	4.6	4.4	4.5
Локація	5.0	4.5	4.3	4.4
Таверна	5.0	4.7	4.6	4.6
Підземелля	4.8	4.3	4.1	4.2
Квест	4.9	4.4	4.2	4.3
Артефакт	5.0	4.5	4.5	4.6

Загальні результати тестування підтвердили стабільну роботу системи, відповідність вимогам та коректну взаємодію компонентів. Виявлені під час тестування граничні випадки (у парсингу OpenAI-відповідей) оперативно усунено.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було:

1. Досліджено предметну область настільно-рольових ігор та проблематику підготовки ігрового контенту майстром гри. Підтверджено актуальність задачі автоматизації: середній МГ витрачає від 3 до 8 годин на підготовку однієї сесії. Ринок HPI демонструє стабільне зростання 10–15% щорічно.

2. Проведено огляд та порівняльний аналіз існуючих рішень (World Anvil, Donjon, Perchance та аналогів). Виявлено ключові недоліки: відсутність інтегрованої AI-генерації якісного тексту, брак персистентності та гнучкості параметризації. Визначено конкурентні переваги розроблюваної системи.

3. Спроєктовано п'ятирівневу клієнт-серверну архітектуру системи з використанням React.js, Node.js, MongoDB та OpenAI API. Архітектурні рішення базуються на принципах Clean Architecture, Feature-Sliced Design та Layered Architecture, що забезпечує незалежність кожного рівня.

4. Розроблено алгоритмічне забезпечення системи: алгоритм формування оптимальних промптів для GPT-4 з урахуванням параметрів генерування, патерни Retry та Circuit Breaker для надійної взаємодії з OpenAI API, механізм SSE-стрімінгу, алгоритм управління витратами через кешування та динамічний вибір моделі.

5. Спроєктовано інформаційне забезпечення, розроблено схему бази даних із таблиць users, generations, templates та sessions; визначено стратегії індексування; реалізовано Redis-кеш, реалізовано в MongoDB.

6. Реалізовано веб-орієнтовану ІС з підтримкою шести типів ігрового контенту (NPC, локації, таверни, підземелля, квести, артефакти) з розширеною параметризацією. Розгортання здійснено засобами Docker Compose.

7. Проведено тестування (модульне, інтеграційне, функціональне, навантажувальне). Навантажувальне тестування підтвердило стабільну роботу при 50 одночасних користувачах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Brown T. B., Mann B., Ryder N., Subbiah M., Kaplan J., Dhariwal P., Neelakantan A., Shyam P., Sastry G., Askell A., Agarwal S., Herbert-Voss A., Krueger G., Henighan T., Child R., Ramesh A., Ziegler D. M., Wu J., Winter C., Hesse C., Chen M., Sigler E., Litwin M., Gray S., Chess B., Clark J., Berner C., McCandlish S., Radford A., Sutskever I., Amodei D. Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems. 2020. Vol. 33. P. 1877–1901.

2. World Anvil. URL: worldanvil.com (дата звернення: 14.05.2026).

3. Donjon. URL: http://donjon.bin.sh/ (дата звернення: 14.05.2026).

4. OpenAI. GPT-4 Technical Report. 2023. URL: https://openai.com/research/gpt-4 (дата звернення: 10.05.2026)

5. Shaker N., Togelius J., Nelson M. J. Procedural Content Generation in Games. Springer, 2016. 246 с.

6. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall, 2017. 432 с.

7. ICv2 & Circana. Hobby Games Market Report 2023. URL: https://icv2.com/articles/markets/view/56589/hobby-game-sales-eke-out-tough-win-2023 (дата звернення: 13.05.2026)

8. Shea M. How Long Does it Take You to Prep Your D&D Game? Sly Flourish. 2021. URL: https://slyflourish.com/how_long_to_prep.html

9. Perchance URL: https://perchance.org/welcome (дата звернення: 13.05.2026)

10. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley, 2002. 560 с.

11. Chodorow K. MongoDB: The Definitive Guide. 3rd ed. O'Reilly Media, 2019. 514 с.

12. Kleppmann M. Designing Data-Intensive Applications. O'Reilly Media, 2017. 590 с.

13. Sommerville I. Software Engineering. 10th ed. Pearson, 2015. 816 с.

14. White J., Fu Q., Hays S., Sandborn M., Olea C., Gilbert H., Elnashar A., Spencer-Smith J., Schmidt D. C. A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. arXiv:2302.11382. 2023.

5. Liu P., Yuan W., Fu J., Jiang Z., Hayashi H., Neubig G. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. ACM Computing Surveys. 2023. Vol. 55, No. 9. Article 195. 35 p.

16. Richardson L. RESTful Web APIs: Services for a Changing World. O'Reilly Media, 2013. 406 с.

17. JWT. Introduction to JSON Web Tokens. URL: https://jwt.io/introduction (дата звернення: 15.05.2026)

18. React Documentation. Managing State. Meta Open Source. URL: https://react.dev/learn (дата звернення: 15.04.2026)

19. Express.js Documentation. URL: https://expressjs.com/ (дата звернення: 15.04.2026)

20. Mongoose Documentation. URL: https://mongoosejs.com/docs/ (дата звернення: 15.04.2026)

21. Docker Documentation. Docker overview. URL: https://docs.docker.com/get-started/overview/ (дата звернення: 15.04.2026)

22. Redis Documentation. Caching. URL: https://redis.io/docs/ (дата звернення: 15.04.2026)

23. Yablonski J. Laws of UX: Using Psychology to Design Better Products. O'Reilly Media, 2024. 164 с.

24. ISO 9241-210:2019. Ergonomics of human-system interaction. Human-centred design. ISO, 2019.

25. Гладун О. М. Основи веброзробки: архітектура клієнт-серверних застосунків. Київ: Академія, 2021. 312 с.

26. Буката Л. М., Мороз І. В. Штучний інтелект та машинне навчання. Львів: ЛП, 2023. 420 с.

27. Задорожна Н. Т., Кузнецова Т. В. Безпека вебзастосунків. Київ: ІПС НАН України, 2022. 248 с.

28. Литвиненко В. І., Тарасенко В. П. Проєктування та розробка баз даних. Вінниця: ВНТУ, 2022. 286 с.

29. Wizards of the Coast. Dungeons & Dragons Player's Handbook, 5th ed. Wizards of the Coast LLC, 2014. 316 с.

30. Paizo Inc. Pathfinder Core Rulebook, 2nd ed. Paizo Inc., 2019. 638 с.

31. TanStack Query v5. URL: https://tanstack.com/query/latest (дата звернення: 22.04.2026)

32. N-tier architecture style. Microsoft Learn. 2025. URL: https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/n-tier (дата звернення: 15.04.2026)

33. Cross-Origin Resource Sharing (CORS). MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS (дата звернення: 15.04.2026)

34. Artillery.io. Load Testing. URL: https://www.artillery.io/docs (дата звернення: 15.05.2026)

35. Jest. JavaScript Testing Framework. URL: https://jestjs.io/ (дата звернення: 15.05.2026)

36. Feature-Sliced Design. URL: https://feature-sliced.design/ (дата звернення: 15.05.2026)

37. Утюк А.Р., Турченко І.В. Веб-орієнтований додаток для генерування контенту для настільних рольових ігор. Scientific Progress: Theories, Applications and Global Impact: матеріали 4-ї Міжнародної науково-практичної конференції (Брага, Португалія, 8–10 червня 2026 р.). Брага, 2026. С. 333-336.

38. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп’яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп’ютерні науки» спеціальності 122 «Комп’ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

Додаток А

Програмний код

Сервіс генерування контенту (GenerationService):

```
// services/generationService.js

const OpenAI = require('openai');
const { buildPrompt, SYSTEM_PROMPTS } = require('../prompts');

class GenerationService {

  async generate(contentType, params, userId) {

    const systemPrompt = SYSTEM_PROMPTS[contentType];
    const userPrompt = buildPrompt(contentType, params);

    const completion = await openai.chat.completions.create({
      model: 'gpt-4',
      messages: [
        { role: 'system', content: systemPrompt },
        { role: 'user', content: userPrompt }
      ],
      temperature: 0.85,
      max_tokens: 1500
    });

    const content =
      JSON.parse(completion.choices[0].message.content);

    return GenerationRepository.save({
```

```
      userId,
      contentType,
      content
    });
  }
}

module.exports = new GenerationService();
```

Mongoose-схема колекції generations:

```
// models/Generation.js

const generationSchema = new mongoose.Schema({

  userId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User'
  },

  contentType: {
    type: String,
    enum: [
      'npc',
      'location',
      'tavern',
      'dungeon',
      'quest',
      'artifact'
    ]
  },

  params: mongoose.Schema.Types.Mixed,
```

```
  content: mongoose.Schema.Types.Mixed,

  isFavorite: {
    type: Boolean,
    default: false
  },

  createdAt: {
    type: Date,
    default: Date.now
  }
});

generationSchema.index({
  userId: 1,
  contentType: 1,
  createdAt: -1
});

module.exports =
  mongoose.model('Generation', generationSchema);
```

Компонент форми генерування контенту:

```
// GeneratorForm.jsx

import { useState } from 'react';

export const GeneratorForm = () => {

  const [params, setParams] = useState({
    contentType: 'npc',
```

```
    setting: 'fantasy',
    system: 'dnd5e'
  });

  return (
    <div>

      <select
        value={params.contentType}
        onChange={(e) =>
          setParams({
            ...params,
            contentType: e.target.value
          })
        }
      >
        <option value="npc">NPC</option>
        <option value="quest">Квест</option>
        <option value="location">Локація</option>
      </select>

      <button>
        Згенерувати контент
      </button>

    </div>
  );
};
```

Додаток Б

Копія опублікованих результатів