ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ
КАФЕДРА ПРОГРАМНИХ ЗАСОБІВ І ТЕХНОЛОГІЙ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи

бакалавра
(освітньо-кваліфікаційний рівень)

на тему *Розробка мультимедійної інтерактивної платформи з історії розробки ігрових движків*

Виконав: здобувач *4* року навчання
групи *4ПР1*
напряму підготовки (спеціальності)
121-Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Полікарпов Дмитрій Андрійович
(підпис, прізвище та ініціали)
Керівник *Огнєва О.Є.*
(підпис, прізвище та ініціали)
Рецензент *Корніловська Н.В.*
(прізвище та ініціали)
Нормоконтролер *Комісаров О. С.*
(підпис, прізвище та ініціали)

Хмельницький – 2026

ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРЗВТЕТ

Інститут, факультет, відділення *Інформаційних технологій та дизайну*
Кафедра, циклова комісія *Програмних засобів і технологій*
Освітньо-кваліфікаційний рівень *бакалавр*
Освітньо-професійна підготовка *Програмна інженерія*
(шифр і назва)
Спеціальність *121-Інженерія програмного забезпечення*
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри *програмних засобів і технологій*
_____ О.Є. Огнєва
«___» _____ 2026 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

Полікарпову Дмитрію Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) *Розробка мультимедійної інтерактивної платформи з історії розробки ігрових движків*

керівник проєкту (роботи) *к.т.н. доцент Огнєва О.Є.*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «*16*» *січня* 202*6* року №*82-с*

2. Строк подання здобувачем проєкту(роботи) *10.06.2026*

3. Вихідні дані до проєкту (роботи) *літературні та періодичні джерела, матеріали переддипломної практики*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз предметної галузі та постановка задачі
2. Проєктування програмного забезпечення
3. Розробка програмного забезпечення
4. Тестування та впровадження програмного забезпечення

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів проєкту (роботи)

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

7. Дата видачі завдання *16.01.2026*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів проєкту (роботи)	Примітка
1	Отримання завдання	16.01.2026	Виконано
2	Підбір літератури	16.01.2026-10.02.2026	Виконано
3	Аналіз предметної області	10.02.2026-27.03.2026	Виконано
4	Розробка та обґрунтування завдання	03.02.2026-29.03.2026	Виконано
5	Розробка концептуальної моделі	20.03.2026-10.04.2026	Виконано
6	Моделювання та проєктування системи	21.03.2026-03.04.2026	Виконано
7	Моделювання та проєктування бази даних	05.03.2026-10.04.2026	Виконано
8	Розробка інтерфейсу та тестування сайту	07.04.2026-20.04.2026	Виконано
9	Оформлення пояснювальної записки	28.04.2026-01.06.2026	Виконано
10	Захист кваліфікаційної роботи	24.06.2026	Виконано

Здобувач _____ *Д.П.Полікарпов*
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи) _____ *О.Є.Огнєва*
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Кваліфікаційна робота бакалавра: 86 сторінок, 34 рисунків, 2 таблиць, 21 джерел, 2 додатків.

Мета роботи: мета кваліфікаційної роботи бакалавра - створити платформу з історії розробки ігрових движків, основним завданням якої є реалізація створення, редагування, видалення статей та представленні їх у потрібному табличному вигляді, причому інформація може бути заснована не на статичних даних, а на шляху до цих даних, що дозволяє автоматично оновлювати їх у разі зміни на оригінальних сайтах. Створити базу даних та структуру таблиць. Заповнити таблиці даними, створити запити на додавання, оновлення та видалення, створити інтуїтивно зрозумілий інтерфейс користувача.

Об'єкт дослідження: веб-орієнтована система створення статей з історії розробки ігрових движків, що функціонує як клієнт-серверна архітектура з відкритим програмним інтерфейсом.

Предмет дослідження: проєктування та розробка інтерактивної платформи з історії розробки ігрових движків. Предметом дослідження є створення, редагування та видалення статей, а також панель керування базами даними чека адміністратора.

Інструменти та засоби розробки: Для розробки інтерактивної платформи з історії розробки ігрових движків було використано редактор коду Visual Studio Code, на реляційній системі керування базами даних MySQL, локальному сервері XAMPP, а також мови програмування HTML (англ. Hyper Text Markup Language - мова розмітки гіпертексту), CSS (англ. Cascading Style Sheets - каскадні таблиці стилів), PHP (англ. Hypertext Preprocessor - гіпертекстовий препроцесор).

Результати роботи: результатом роботи є мультимедійна інтерактивна платформа з історії розробки ігрових движків , яка дозволяє додавати, редагувати та видаляти статті (якщо у джерелі змінилися дані і потрібно, щоб

поточні дані були актуальними, при натисненні на відповідну кнопку, дані автоматично завантажуються заново). Проведено системне тестування, що підтвердило відповідність програмного продукту технічним вимогам.

Новизна рішення: розроблений вебдодаток має не тільки теоретичне значення, розвиваючого теорію проєктування вебдодатків, а і практичне значення і може сприяти підвищенню якості аналітичної роботи, яка може бути використана, як основа для повнофункціональної платформи з історії розробки ігрових движків з предметної області. Запропонований вебдодаток можна буде використовувати для подальшого удосконалення цієї методики. Web додаток пройшов ряд тестових випробувань на предмет виявлення помилок у проєктуванні та реалізації. Перевірка підтвердила відповідність вимогам функціональності, надійності, мобільності та зручністю використання. Наведені перспективи розвитку.

КЛЮЧОВІ СЛОВА: HTML, CSS, PHP, ВЕБ-СЕРВЕР, БАЗА ДАНИХ, MYSQL, VISUAL STUDIO CODE, XAMPP.

АНОТАЦІЯ

Полікарпов Дмитрій Андрійович створення мультимедійної інтерактивної платформи з історії розробки ігрових движків - Кваліфікаційна робота бакалавра. Херсонський національний технічний університет, Хмельницький, 2026 р.

Перший розділ «Аналіз предметної області» містить огляд створення ефективного серверного прикладного інтерфейсу, який стане основою цифрової платформи для створення статей. У розділі було проведено аналіз предметної області, визначено основні категорії користувачів та функціональні сценарії їхньої взаємодії із системою.

Другий розділ «Проєктування програмного забезпечення» складається з таких підрозділів: «Архітектура вебдодатку», «Побудова діаграм варіантів», «Проєктування структури даних», «Проєктування поведінки», «Проєктування інтерфейсу» та «Висновки до розділу». Спроєктовано архітектуру програмної системи: розроблено UML-діаграми варіантів використання, класів, послідовностей, розгортання, компонентів та ER model. Обґрунтовано вибір технологічного стеку.

Третій розділ «Розробка програмного забезпечення» складається з п’яти підрозділів: «Середовище та інструменти розробки», «Реалізація ключових компонентів системи», «Реалізація бази даних та взаємодія з нею», «Програмування прикладного API» та «Висновки до розділу». Реалізовано програмний продукт засобами, а саме: веб-сервер XAMPP, база даних MySQL, бек-енд PHP, мова стилю сторінок CSS, HTML – інструментарій для браузера. Проведено модульне та інтеграційне тестування з покриттям коду 100 %. Розроблена інструкція користувача.

Четвертий розділ «Тестування та впровадження програмного забезпечення» складається з шести підрозділів: «Модульне тестування», «Інтеграційне та UI-тестування», «Аналіз результатів тестування», «Інструкція користувача», «Впровадження та розгортання» та «Висновки до розділу».

Результати даної роботи можуть бути використані, як основа для повнофункціональної платформи з історії розробки ігрових движків з подальшою інтеграцією користувальницького та адміністративного частини. Практична значущість полягає у інформаційному просторі.

ABSTRACT

Polikarpov Dmitrii Andriyovych Development of a multimedia interactive platform on the history of game engine development — Bachelor qualification work. Kherson national technical university, Khmelnytskyi 2026 year.

The first chapter «Analysis of the Subject Area» provides an overview o of creating an effective server-side application interface, will become the basis of a digital platform for creating articles. The section analyzed the subject area, identified the main categories of users and functional scenarios of their interaction with the system.

The second chapter «Desing software» consists of the following subsections: «Web application architecture», «Development of Use Case Diagram», «Data structure design», «Development of Behavior», «Development of Interface» and «Conclusions to the section». Designing a web application architecture: the implemented UML - use case diagram, class diagram, sequence diagram, deployement diagram, component diagram and ER model. Reasonable choose technical stack.

The third chapter «Development of software» consists of five sections: «Development environment and tools», «Implementation of key system components», «Database implementation and interaction», «Application API programming» and «Conclusions to the section». The implemented of software. Namely: Apache web server; MySQL database, PHP back and end, page style language CSS, HTML - browser tools. Comprehensive unit and integration testing with code coverage 100 %. Development of the user manual.

The fourth chapter «Testing and Implementation of software» consists of six sections: «Module testing», «Integration and UI testing», «Analysis of test results», «User manual», «Implementation and deployment» and «Conclusions to the section». The result of the work is a fully functional backend system, ready for deployment and capable of serving as the foundation for a job search platform with future user and admin side integration. The practical consists of information space.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

АСОІ - автоматизована система обробки інформації.

АСУ - автоматизована система управління.

ІТ - інформаційні технології.

ПД - програмна документація.

ПП - програмний продукт.

ПЗ – програмне забезпечення

СУБД - система управління базами даних.

БД - бази даних

VSC - Visual Studio Code.

HTTP - HyperText Transfer Protocol – протокол передачі гіпертексту.

HTML - Hyper Text Markup Language – стандартизована мова розмітки документів для перегляду вебсторінок у браузері.

CSS - Cascading Style Sheets – мова стилю сторінок.

PHP - Hypertext Preprocessor – скриптова мова програмування, розроблений для веб-сторінок і інтерпретується вебсервером у HTML-код.

UI - User Interface – інтерфейс користувача.

AD - Activity Diagram – діаграма діяльності.

IDE - Integrated Development Environment – інтегроване середовище розробки.

DBMS - Database Management System – система керування базами даних.

CRUD - Create, Read, Update, Delete – основні операції над даними.

CI/CD - Continuous Integration / Continuous Delivery – безперервна інтеграція та доставка.

URL - стандартизована адреса певного ресурсу в інтернеті.

MySQL - система управління реляційною базою даних з відкритим кодом.

XAMPP - локальний вебсервер з відкритим початковим кодом, що містять модулі HTTP-сервер Apache, базу даних MySQL, FTP-менеджер FileZila, система передачі пошти та вебсервер Tomcat.

ВСТУП

У сучасних умовах поширення інтернетизації суспільства, питання ефективного створення статей набуває все більшого значення. Зі зростанням кількості створених статей, динамічністю зміни вимог та високим темпом розвитку IT-сектору, класичні методи створення статей через газети, оголошення або рекрутингові агентства поступово втрачають свою ефективність. На їх заміну приходять автоматизовані цифрові платформи, які забезпечують інтерактивну взаємодію між користувачами та адміністраторами.

Найбільш зручним і масштабованим підходом до створення таких платформ є побудова клієнт-серверних інформаційних систем. У цьому контексті розробка мультимедійної інтерактивної платформи з історії розробки ігрових движків є ключовим завданням, оскільки саме цей компонент визначає структуру даних, доступ до ресурсів, механізми безпеки та правила взаємодії з користувальницькими застосунками.

Актуальність теми. Дослідження зумовлено необхідністю створення відкритого, гнучкого і добре документованого серверного інтерфейсу, що дозволяє реалізувати платформу з підтримкою ролей користувачів та адміністраторів, аналітикою та інтегрованою системою повідомлень. Крім того, архітектура платформи дозволяє легко масштабувати систему CRM-системами, а також забезпечувати її безперебійну роботу в умовах високого навантаження.

Об’єкт дослідження. Об’єктом роботи є повнофункціональна серверна частина інформаційної системи, що реалізована з використанням інструментарію: Visual Studio Code - редактора коду, XAMPP - локального вебсервера, мови HTML - мови розмітки гіпертексту, CSS - каскадних таблиць стилів і PHP - гіпертекстового препроцесора.

Предмет дослідження. Проєктування та розробка інтерактивної платформи з історії розробки ігрових движків.

Мета дослідження. Мета кваліфікаційної роботи бакалавра – розробка серверного прикладного програмного інтерфейсу для інформаційної системи з історії розробки ігрових движків. Предметом дослідження виступає проєктування,

реалізація та тестування мультимедійної інтерактивної платформи згідно з вимогами до функціональності, безпеки, масштабованості та підтримуваності.

Задачі дослідження. Для досягнення цієї мети в роботі вирішуються наступні задачі:

- проведення аналізу предметної області та постановки задачі;
- моделювання функціональних процесів і вимог до системи;
- побудова архітектури програмного забезпечення;
- розробка платформи згідно з обраними технічними засобами;
- реалізація реєстрацію користувачів, створення, перегляд, редагування, статті, редагування профілів користувачів, адміністративний контроль;
- проведення інтеграційного та системного тестування;
- підготовка інструкцій щодо встановлення, запуску та експлуатації програмного продукту.

Методи дослідження - аналіз та узагальнення інформації про існуючі веб-ресурси та автоматизації подібних до предметної галузі платформи з історії розробки ігрових движків (їх структури та особливостей); теоретичний аналіз та узагальнення інформаційних джерел з тематики створення вебресурсів, моделювання та проєктування баз даних.

Практична значущість. Інформаційна діяльність передбачає роботу з інформацією, її обробку, прийняття рішення з аналізу тієї чи іншої ситуації, що відноситься до проблеми, тематичну обробку інформації, підготовку та створення статей.

Структура. Кваліфікаційна робота бакалавра складається з 86 сторінок, 34 рисунків, 2 таблиць, 2 додатки, 21 джерел.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Опис та загальна характеристика предметної області

Сучасне життя дедалі активніше адаптується до нових технологічних реалій, інтернет стає одним із ключових технічних інструментів сучасної людини і саме в цьому контексті виникає потреба у створенні нових статей, де модороби та розробники можуть знайти потрібну інформацію про розробки конкретного ігрового движка [11].

Предметна область, яка є об'єктом дослідження у межах цієї кваліфікаційної роботи, сформована в контексті актуальних статей, підтримкою та вдосконаленням взаємодії між основними його учасниками - користувачами та адміністративним персоналом. У фокусі уваги - створення ефективного серверного прикладного інтерфейсу, який стане основою цифрової платформи для створення статей.

Розглянута предметна область уособлює функціональну модель типової онлайн-системи з історії розробки ігрових движків, де кожна дія користувача, кожен запит або відповідь мають бути оброблені, збережені, пов’язані з іншими подіями та, у разі необхідності, використані в рамках прийняття рішень. Як бізнес-система, така платформа розглядається як комплекс взаємопов’язаних процесів, що забезпечують досягнення головної мети – створення статей, швидкого та прозорого процесу публікації з мінімальними часовими та адміністративними витратами.

У сучасних умовах цифрової трансформації традиційні способи створення статей втрачають свою ефективність, поступаючись місцем високотехнологічним рішенням, які краще відповідають вимогам інформаційного простору [14]. Зокрема, цифрові сервіси інформації все активніше замінюють застарілі канали комунікації між читачами та інформаторами і забезпечуючи миттєвий обмін інформацією. При розробці

інформаційної структури платформи потрібно знати, що у залежності від кількості інформаційних матеріалів, які потрібно розмістити на сайті, модель сайту може бути лінійна, гібридна, мережева та ієрархічна.

В інтернет просторах такі платформи, як Wikipedia, Grokipedia, Scholarpedia, Fandom вже сформували стійку екосистему, що функціонує як практичне втілення принципів інформаційних ресурсів. Застосування технологічних рішень у цих системах свідчить про успішну інтеграцію ключових функцій - від публікації статті до аналітики ефективності взаємодії між учасниками. Вони демонструють успішну інтеграцію базових функцій пошуку інформації - створення, редагування видалення статей, ведення комунікації – у масштабовані онлайн-інструментів, доступні з будь-якої точки світу.

У логічному продовженні тенденцій, описаних вище, слід відзначити зміну поведінкових запитів користувачів, що безпосередньо впливають на вимоги до інформаційних систем. Розвиток дистанційної, гібридної, проєктної та фриланс-орієнтованої шукачів інформації, а також зростання кількості читачів у інформаційному просторі, модороби та розробники формують зовсім інший рівень очікувань від онлайн-платформ.

Усе це зумовлює підвищені вимоги до гнучкості, масштабованості, стабільності й адаптивності систем, які забезпечують інформаційну взаємодію на інформаційному просторі. У таких умовах публікація статті – як з боку інформатора, так і з боку користувачів – стає не лише фактором оптимізації, а необхідною умовою функціонування конкурентоспроможної онлайн-платформи. Забезпечення мультимедійної інтерактивної платформи з історії розробки ігрових движків, двосторонньої комунікації та мінімізації ручної обробки даних є ключовими перевагами платформи як архітектурного рішення для підтримки таких вимог.

Враховуючи ці тенденції, розроблена система покликана задовольнити зростаючі вимоги до швидкодії, інтерактивності та персоналізації сервісу з історії розробки ігрових движків. Вона реалізує ключові функції, необхідні для

забезпечення ефективної взаємодії між користувачами: формування профілів, пошук необхідної інформації, а також забезпечення зручної та оперативної комунікації між сторонами.

Функціональна логіка системи базується на клієнт-серверній архітектурі, у платформи виконує роль центрального компонента для обміну даними. Вибір архітектурного стилю для побудови прикладного програмування інтерфейсу відбувався на основі каскадних таблиць стилів. Обидва мають свої переваги. Зокрема, платформа дозволяє користувачу самостійно створювати статті. Проте, у контексті даної платформи, був обраний як більш стабільний, передбачуваний і стандартизований підхід, який спрощує документування через CSS та полегшує інтеграцію з кешуванням, логуванням та безпекою на основі JWT [20]. Крім того, інтерактивна платформа краще підходить для CRUD-орієнтованих платформ, таких як система з пошуку інформації з історії розробки ігрових движків. Через UI реалізується більшість бізнес-процесів: реєстрація нових користувачів, авторизація користувачів, збереження та редагування профілів, сторінка публікації статей, сторінка редагування статей і панель керування базою даних.

Такий підхід забезпечує структурованість інформаційних потоків, ізоляцію логіки бізнес-операцій, а також спрощує інтеграцію з іншими цифровими сервісами та майбутню масштабованість системи.

До основних учасників цієї системи належать автори статей, користувачі та адміністратори платформи. Користувачі створюють статті, залишають коментарі та спілкуються з автором статті. Адміністративна частина системи забезпечує підтримку стабільності платформи, модерацію контенту, контроль за дотриманням правил, а також відповідає за підтримку доступності та безпеки роботи сервісу.

Особливістю предметної області є необхідність обробки значних обсягів структурованої інформації з високими вимогами до її актуальності та достовірності. Кожна дія користувача породжує певну інформаційну подію, яка має бути зафіксована, оброблена та у разі потреби – використана у базі даних.

Враховуючи масштаби системи та потенційну кількість користувачів, необхідно передбачити високий рівень масштабованості, підтримку одночасних сесій, захист даних, модульність і гнучкість у подальшому розвитку.

Актуальність створення статті у межах розглянутої предметної області обумовлена як зростанням обсягів інформації, що підлягає обробці, так і збільшенням вимог до якості сервісу. Зниження навантаження на персонал та мінімізація людського фактору – усе це є критичними факторами успішної реалізації бізнес-логіки в межах цифрової платформи з історії розробки ігрових движків.

Одним із таких ключових рішень є розробка інтерфейсу прикладного програмування, що забезпечує стандартизований, незалежний від учасників та розширюваний спосіб взаємодії між різними частинами інформаційної системи. Такий підхід дозволяє досягати високого рівня адаптивності до інформаційному просторі та відкритості до інтеграції з іншими сервісами, включаючи зовнішні мобільні застосунки, чат-боти чи корпоративні CRM-рішення [9].

Інтерактивна платформа є одним із найефективніших архітектурних рішень для реалізації масштабованих вебсистем, оскільки воно забезпечує чіткий розподіл логіки, а також дозволяє публікувати, редагувати та видаляти статті.

Таким чином, розглянута предметна область характеризується як сукупність інформаційно насичених процесів взаємодії між учасниками та середовище де можна знайомитись із готовими статтями, створювати нові, редагувати, та видаляти вже створені статті. Актуальність платформи полягає в необхідності покращення ефективності, продуктивності та прозорості всіх основних процесів поширювання інформації, що в умовах сучасного інформаційного простору дуже актуально.

1.2. Огляд та аналіз існуючих рішень

Інформація - один з найважливіших активів сучасної організації. Обсяги даних, які зберігають і обробляють сучасні організації, вимірюються сотнями

гігабайт і продовжують зростати. Вартість інформації, що зберігається в комп'ютері в десятки, а то і в сотні разів перевищує вартість обладнання. Навряд чи когось треба сьогодні переконувати в необхідності систем резервного копіювання та відновлення статей. Тепер системи зберігання даних стали широко поширеним товаром, який повинен відповідати потребам користувача, володіти певними споживчими характеристиками і служити інструментом вирішення унікальних завдань щодо збереження інформації [20].

Wikipedia - це унікальний веб сайт, що поєднує в собі засіб збору і каталогізації наукових статей для подальшої підготовки пристатейних списків, і наукову соціальну мережу, яка дозволяє організовувати персональну наукову бібліотеку, спільно працювати над статтями, а також знаходити однодумців, і вивчати тренди сучасних досліджень. Творці Wikipedia - Джиммі Вейлзом та Ларрі Сенгером на рис. 1.1.

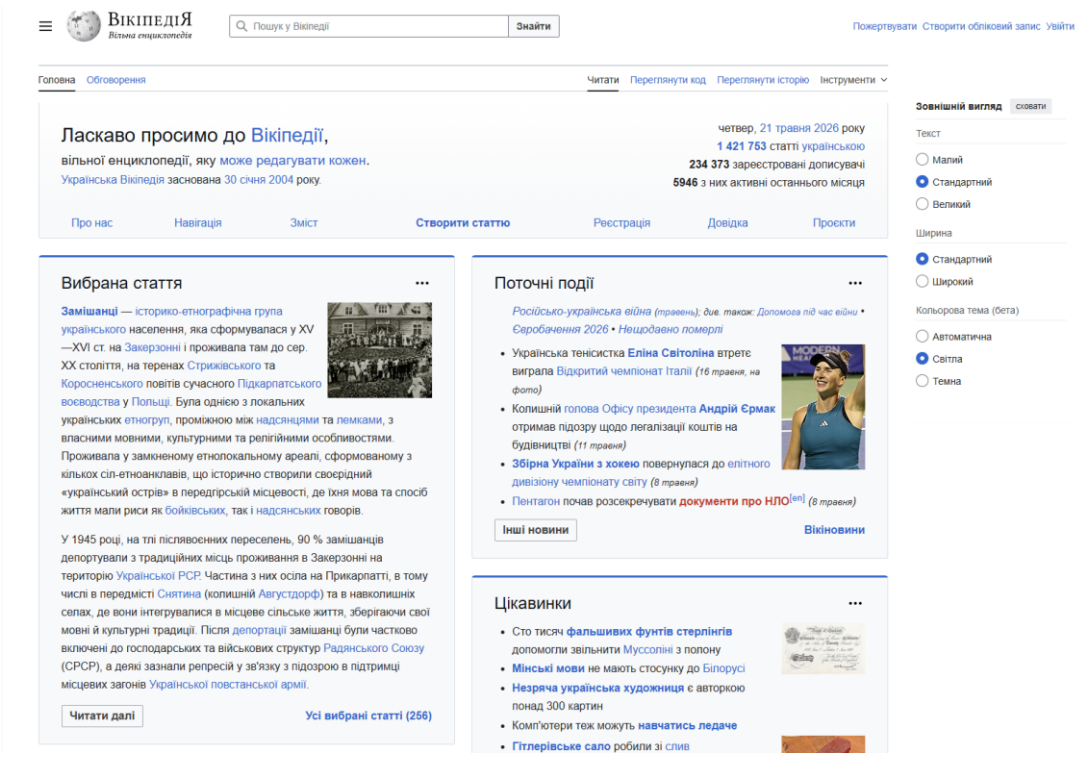


Рисунок 1.1. Інтерфейс веб-сайту Wikipedia

Вебсайт інформацію про статті отримує з он-лайн каталогів. Можна додати і інші, але, як правило, встановлених достатньо. Система пошуку дозволяє шукати по багатьом параметрам, але сам пошук реалізований достатньо не гнучко, на точних збігах, іноді виникають складнощі з грецькими символами, апострофами і тому подібне. Якщо не буде ідентичного збігу, пошуковий запит прийде порожнім. Наприклад, шукати по імені автора, року публікації і назвою журналу, ніж назві статті, або використовувати перші кілька слів назви, замість повної назви.

Fandom - комерційна система управління бібліографічною інформацією, що застосовується для управління посиланнями і бібліографією, що дозволяє відформатувати їх згідно з численними стандартам цитування на рис. 1.2.

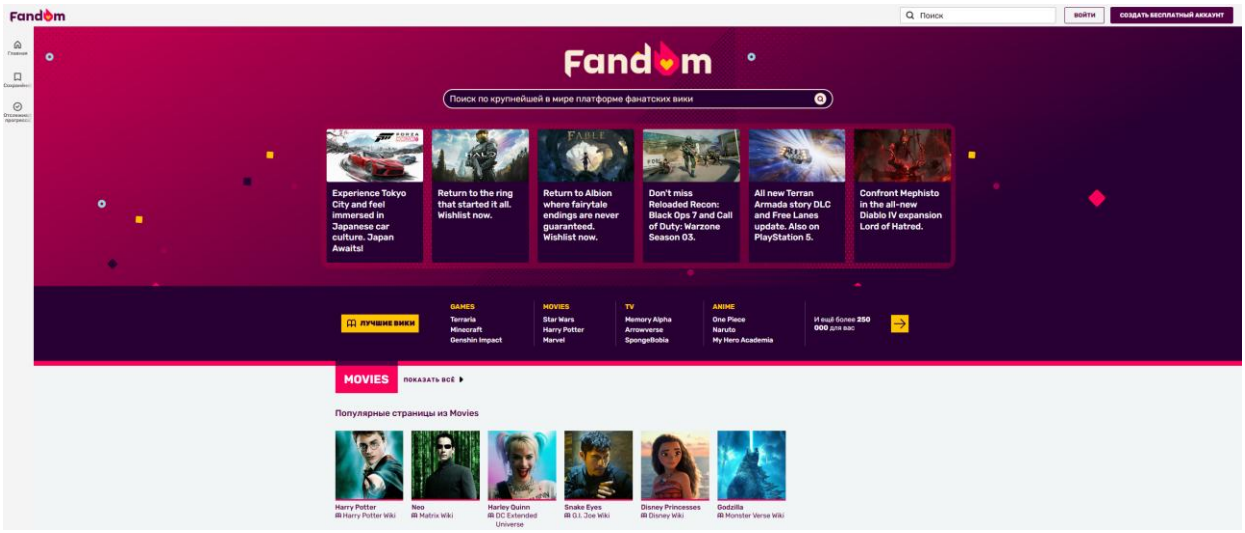


Рисунок 1.2. Інтерфейс веб-сайту Fandom

PCGamingWiki - це сайт з технічної інформації про комп'ютерні ігри та ігровий двіжок (засіб збору і каталогізації статей для подальшої підготовки пристатейних списків), яка дозволяє організовувати персональну наукову бібліотеку, спільно працювати над статтями, а також знаходити однодумців (див. на рис. 1.3).

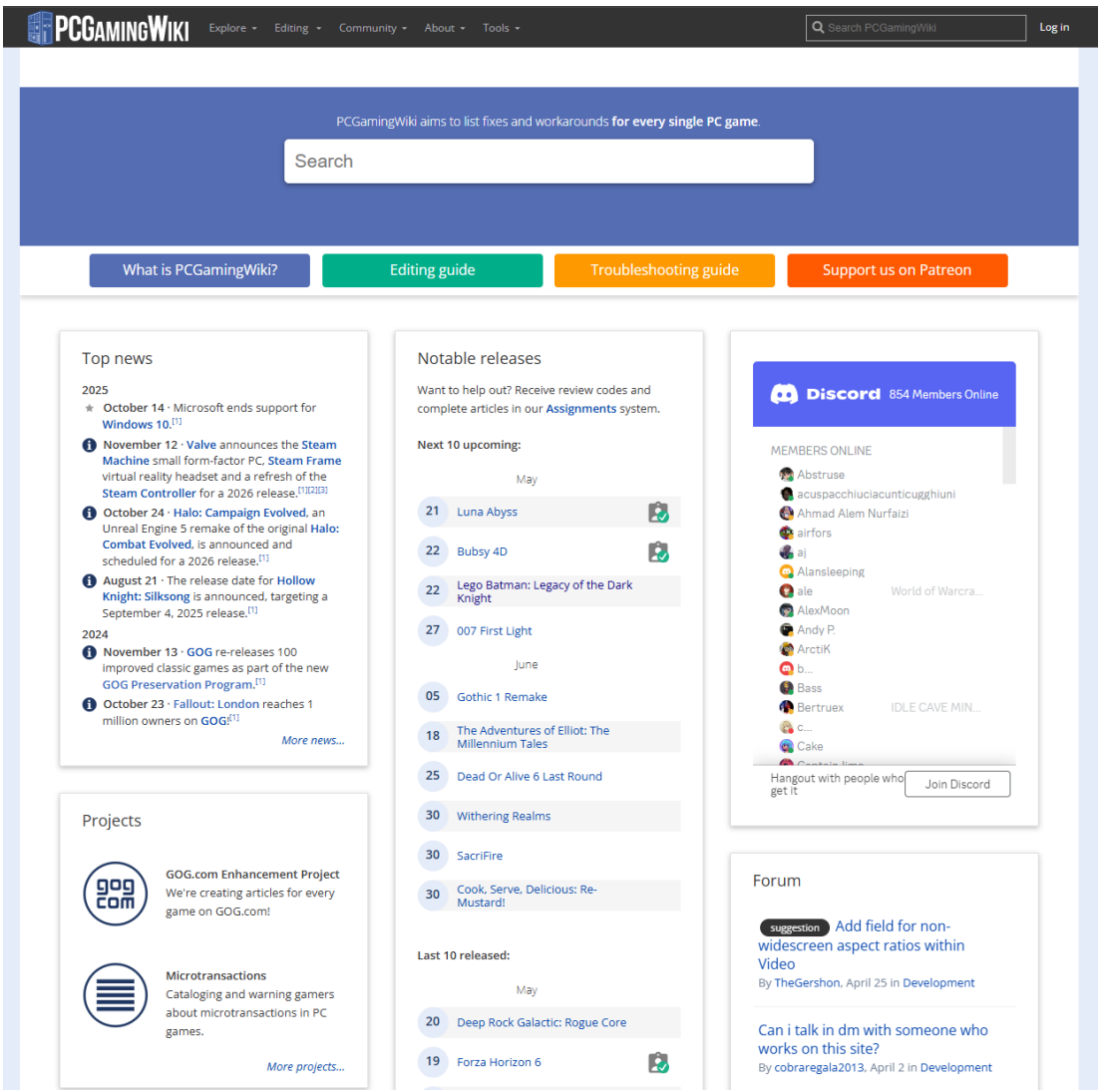


Рисунок 1.3. Інтерфейс веб-сайту PCGamingWiki

Інший вебсайт Wikiwand є альтернативною версією Wikipedia дозволяє стежити за своєю власною сторінкою і впливати на нього безпосередньо з інтерфейсу Wikiwand (див. на рис. 1.4).

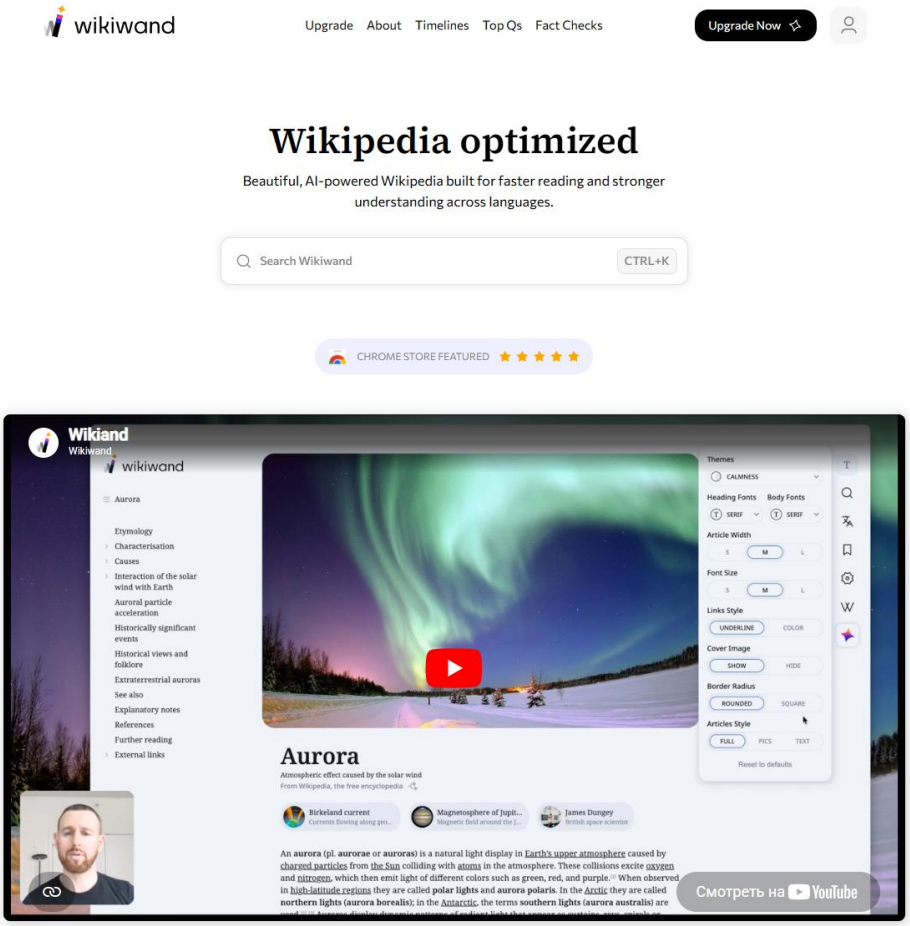


Рисунок 1.4 Інтерфейс веб-сайту PCGamingWiki

Платформа Giant Bomb - база даних з комп'ютерної гри та ігрових двіжків (рис. 1.5).

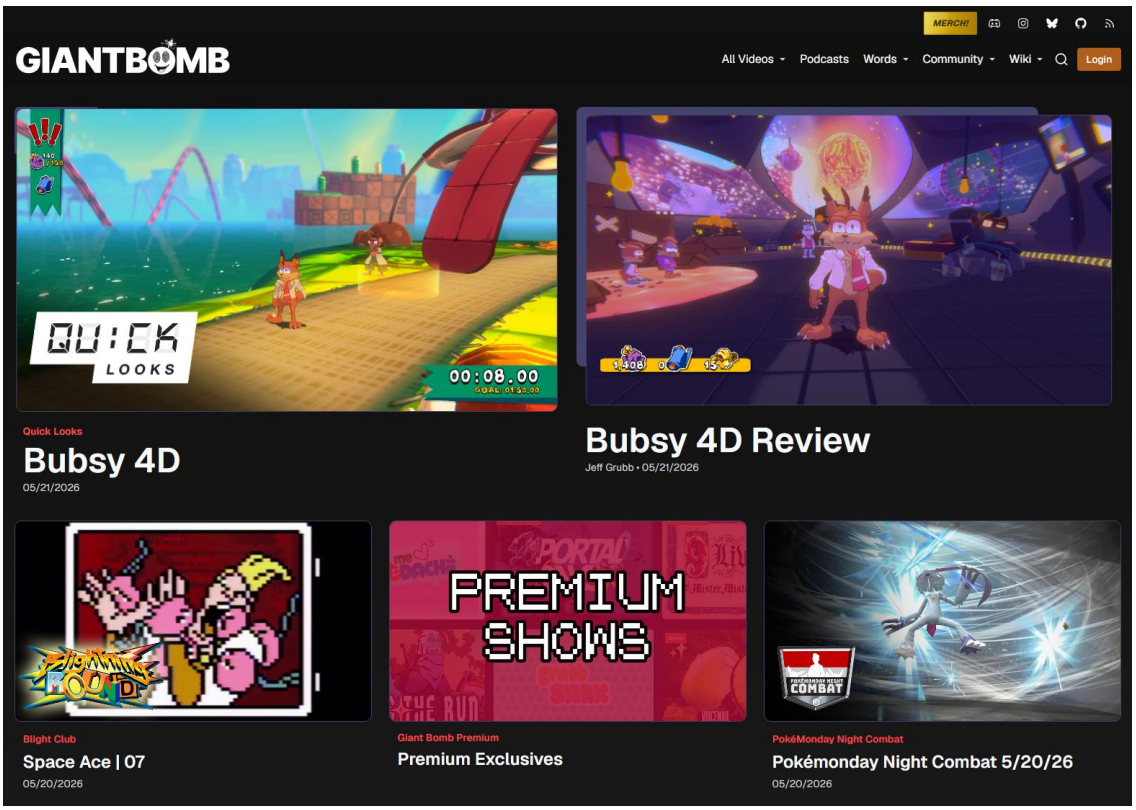


Рисунок 1.5 Інтерфейс веб-сайту Giant Bomb

Для порівняння переваг та недоліків рішень було створено таблицю 1.1.

Таблиця 1.1. Порівняння таблиця з перевагами та недоліками рішень

Назва платформ	Переваги	Недоліки
Wikipedia	Є літературні джерела, безкоштовний ресурс	Не всі статті проходять адміністративну перевірку
Fandom	Є літературні джерела, можливість редагувати та видаляти свої статті	Не всі статті проходять адміністративну перевірку, велика кількість реклами.
PCGamingWiki	Є літературні джерела, э рекомендації щодо налаштування.	Менше інформації.
Wikiwand	Є літературні джерела, зручний інтерфейс та навігація.	Вторинність статей від Wikipedia.
Giant Bomb	Мультимедійний контент	Відсутність літературних джерел, менше інформації

Аналогів розробляємого додатку у результаті аналізу виявлено не було, тому дане дослідження та розробка програмного продукту є актуальною та новою.

Таким чином, запропонований підхід дозволяє розробити платформу з історії розробки ігрових движків, для знаходження потрібної інформації.

1.3. Розробка проєктних специфікацій

Розробка програмного забезпечення на основі мультимедійної інтерактивної платформи з історії розробки ігрових движків вимагає точного формалізованого опису інтерфейсів прикладного програмування, які служать каналом взаємодії між клієнтськими додатками та серверною логікою. В

контексті розробки цифрової платформи для створення статей проєктні специфікації виконують ключову функцію: вони не лише створює статтю, а й редагування та видалення вже існуючих статтей.

У межах цього підрозділу представлений систематизований опис специфікацій платформи, яка реалізує взаємодію між основними ролями: користувачем, адміністратором та зовнішніми сервісами. Кожна специфікація показує логіку відповідного функціонального модуля, структуру запитів і відповідей, формат даних, правила авторизації та обмеження доступу. Формування такого рівня опису дозволяє не лише стандартизувати виклики, але й сприяє надання чинності бізнес-логіки на ранньому етапі проєктування.

На відміну від інтерфейсів користувача, які орієнтовані на візуальну взаємодію, специфікації платформи є внутрішнім орієнтуванням системи, що забезпечує її стабільність та інтеграційну сумісність. У розробленому проєкті передбачено подання цих специфікацій у табличному вигляді, що буде оформлено у відповідних додатках до дипломної роботи. Такий підхід дозволяє забезпечити однозначність інтерпретації та підтримку автоматизованого тестування і генерації документації.

Формалізована структура мультимедійної інтерактивної платформи з історії розробки ігрових движків містить понад десять ключових ендпоінтів, згрупованих за функціональними блоками. Основними напрямами реалізації API є автентифікація, управління обліковими записами, створення профілів, робота із статтями, обробка відгуків, внутрішня комунікація, ведення аналітики та адміністративний контроль. Усі ендпоінти реалізуються відповідно до методології SESSION із застосуванням HTTP методів (GET, POST, PUT, DELETE), авторизаційних токенів.

Для прикладу, у модулі автентифікації реалізовано маршрути для реєстрації користувачів, авторизації, відновлення паролю, а також оновлення токенів. Кожен запит у цьому модулі передбачає перевірку даних та обробку помилок на рівні логіки контролера, з поверненням релевантних кодів (наприклад, 400 у разі помилок валідації або 409 при спробі повторної

реєстрації). На додаток, усі дані користувачів зберігаються у зашифрованому вигляді, що відповідає вимогам до забезпечення інформаційної безпеки.

Важливу роль відіграють специфікації, пов’язані з обробкою інформації про статті. Передбачено повний набір CRUD (create, read, update, and delete) операцій для оновлення, створення, перегляду та видалення вже готових статей. Так, ендпоінт POST дозволяє користувачу створити нову статтю, вказавши назву ігрового двіжка, мову програмування, опис та логотип ігрового двіжка. У відповідь система повертає унікальний ідентифікатор статті. Для запобігання зловживанням реалізовано обмеження: наприклад, не можна створити більше ніж десять активних статей для несертифікованого користувача, а всі публікації проходять автоматичну модерацію на основі ключових слів.

Мною було обрано інструментарій Visual Studio Code редактор коду, і XAMPP локальний вебсервер, і відповідно, мови HTML мова розмітки гіпертексту, CSS каскадні таблиці стилів і PHP гіпертекстовий препроцесор.

Visual Studio Code був випущений 14 квітня 2016 року від компанією Microsoft на платформах Windows, Linux та MacOS. Функції редактору коду включать підтримку налагодження, інтелектуальне завершення коду, підсвічування синтаксису, фрагменти, рефакторинг коду та вбудований Git [5].

XAMPP локальний вебсервер з відкритим початковим кодом, який містить модулі HTTP-сервер Apache, FTP-менеджер FileZila, базу даних MySQL, система передачі пошти та вебсервер Tomcat [6].

HTML (англ. Hyper Text Markup Language - мова розмітки гіпертексту) - стандартизована мова розмітки документів для перегляду веб сторінок у браузері. Елементи HTML складаються з тегами написаними із використанням кутових дужок [3].

CSS (англ. Cascading Style Sheets - каскадні таблиці стилів) - мова стилю сторінок, використовується для синтаксису та опису їхнього зовнішнього вигляду. За допомогою CSS можна вказати такі параметри як розміри, шрифти, кольори, відступи та інші візуальні характеристики елементів інтерфейсу [2].

PHP (англ. Hypertext Preprocessor - гіпертекстовий препроцесор) це скриптова мова програмування, розроблена для вебсторінок і які інтерпретується вебсервером у HTML-код. Скриптова мова програмування виконує всі серверні операції до додавання, видалення, редагування, бази даних та виконання запитів [4].

Для підтримки адміністративного контролю проєктом передбачено ряд ендпоінтів, доступних лише користувачам із роллю адміністратора. Зокрема, модуль аналітики дозволяє здійснювати вибірку за періодами, виводити статистику переглядів, відгуків та конверсії користувачів. Специфікації таких запитів розроблено з урахуванням можливості подальшого підключення бази даних.

У результаті треба додати, що сформовані специфікації платформи становлять повноцінну технічну модель функціонування цифрової інтерактивної мультимедійної платформи з історії розробки ігрових движків. Їх структура забезпечує покриття всіх бізнес-процесів, дозволяє підтримувати безпеку, продуктивність і масштабованість системи, а також аналітичних сервісів. Розроблені специфікації відображають вимоги, які сформульовані на основі логіки сценаріїв використання (usecase моделей), відповідають принципам платформи.

1.4. Постановка задачі та визначення вимог

Метою кваліфікаційної роботи бакалавра є створення вебдодатку, який наддасть можливість користувачам створювати статті. Експлуатація розробленого вебдодатку повинна допомогти швидко зібрати та обробляти інформацію, формуючи її у табличному вигляді.

Для створення платформи з історії розробки ігрових движків розглянемо постановка задачі:

- створити головне меню;
- створити сторінку реєстрації та авторизації;

- створити зрозумілий та чіткий інтерфейс для користувачів та адміністраторів;
- створити сторінку редагування профілю;
- створити сторінку із статями;
- створити сторінку публікації та редагування статей;
- створити панель керування базами даними для адміністраторів.

Додаткові вимоги:

- англійський інтерфейс;
- зручний інтерфейс;
- дизайн сторінок з використанням сучасних стандартів вебдизайну.

Для функціональні вимог додатка була розроблена діаграма прецедентів (Use Case) (див. рис. 1.6) для визначення функціональних вимог майбутнього додатка. Дана діаграма прецедентів розмежовує додаток і його оточення і так само відображає, як користувач взаємодіє з додатком.

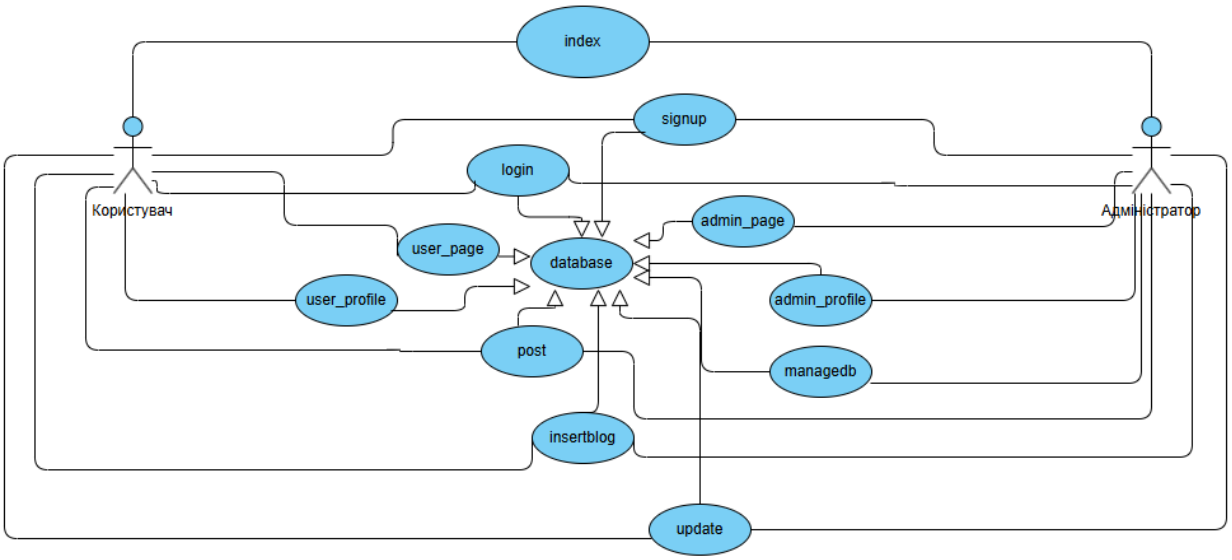


Рисунок 1.6. Діаграма прецедентів

Представлена діаграма показує необхідний функціонал вебдодатку та через які варіанти використання виконується взаємодія користувача та адміністратора з системою.

Поряд з усіма функціональними вимогами платформи є і нефункціональні:

- платформа інтерфейса повинна бути простою і зрозумілою;
- адміністратори повинні нести відповідальність;
- платформа повинна мати можливість скинути пароль;
- система повинна мати захист персональних даних користувачів.

Одним із обмежень користувачів при взаємодії з платформою з історії розробки ігрових движків, повинні використовувати браузер. Для умови інформаційної безпеки, персональні дані користувача будуть знаходиться у базі даних.

Використовуючи визначені функціональні та нефункціональні вимоги створюваного додатку, на даному етапі необхідно побудувати модель логіки роботи програми. Потрібно визначити, що на головному екрані додатка відображається меню, за допомогою якого можна завантажити необхідну сторінку для виконання тієї чи іншої операції над елементами таблиці. Кожна така сторінка містить необхідний об’єм операцій з базою даних та іншими елементами додатку для виконання логічно-обґрунтованого редагування даних в додатку.

1.5. Висновки до розділу 1

У межах першого розділу було проведено системне дослідження предметної області, що охоплює функціонування цифрової платформи з історії розробки ігрових движків як складної бізнес-системи. Було виявлено, що процеси взаємодії між користувачами та адміністраторами відбуваються у межах взаємопов’язаних інформаційних потоків, які потребують чіткої структуризації, регламентованості та автоматизованої підтримки. Робота над розділом передбачала глибоке опрацювання функціональних і нефункціональних характеристик системи, визначення ролей користувачів.

Під час аналізу були виявлені численні недоліки поточної моделі, а саме: обмеженість у масштабуванні, відсутність наскрізної платформи архітектури,

недостатній рівень автоматизації окремих процесів, недостатня підтримка ролей і нестача гнучких засобів взаємодії між сторонами.

Також у даному розділі були розглянуті засоби та технології для створення вебдодатку. Вони були подані у табличному форматі, згідно до прийнятих практик специфікації вимог, і стали базою для подальшого проєктування архітектури та реалізації логіки програмного забезпечення. У результаті була спроєктована повноцінна логічна модель системи, яка дозволяє не лише ефективно реалізувати її у коді, але й забезпечити її масштабованість, тестованість і придатність до інтеграції.

До особливостей розробляємого вебдодатку відноситься ще й те, що за його допомогою можна постійно отримувати актуальні дані і даний підхід можна розповсюдити на інші домени, коли необхідно утримувати дані актуальними.

У підсумку було поставлена мета проєктування нового серверного рішення, яке дозволить реалізувати автоматизовану обробку основних процесів системи. Сформульовано перелік інформаційних задач, які потребують модернізації, а також визначено задачу і зміст технічного проєктування програмного продукту, що буде реалізовано у наступних розділах пояснювальної записки. На основі цього розділу була побудована архітектурна логіка, визначені інтерфейси, а також сформована основа для етапів розробки, тестування та впровадження програмного продукту.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Архітектура веб-додатку

Прогрес інформаційних технологій привів до великого зростання об’ємів інформації, серед якої потрібно створювати, редагувати та видаляти статті.

Тому особливо актуально зараз стоїть питання проєктування та розробки вебдодатків, які мають стати основою для створення статей. Для цього необхідно реалізувати теоретично - обґрунтовані технології, які охоплюють всі ланцюги роботи з інформацією, особливо її агрегації. Представляється дуже важливим, щоб платформи з історії розробки ігрових движків були адаптованими, тобто орієнтованими на інформаційні потреби користувачів.

Діаграма компонентів (Component diagram) - шаблон проєктування, який використовується в більшості для відображання залежності між компонентами програмного забезпечення. Ця діаграма показує тільки структурні характеристики, а для відображення окремих екземплярів компонент потрібно використовувати діаграму розгортання.

Діаграми компонентів використання дозволяють [7]:

- компоненти взаємодіють між собою через чітко визначені зв’язки;
- побудований інтерфейс постачання дозволяє класам реалізувати цей інтерфейс;
- висувати вимоги класам, що залежать від даного інтерфейсу.

Діаграма компонентів (рис. 2.1) відображає структурну організацію програмного додатку як сукупності модулів, які відповідають за реалізацію окремих підсистем. У межах побудованої архітектури платформи для створення статей система поділена на низку логічно-відокремлених компонентів, кожен з яких реалізує визначений набір функціональних вимог.

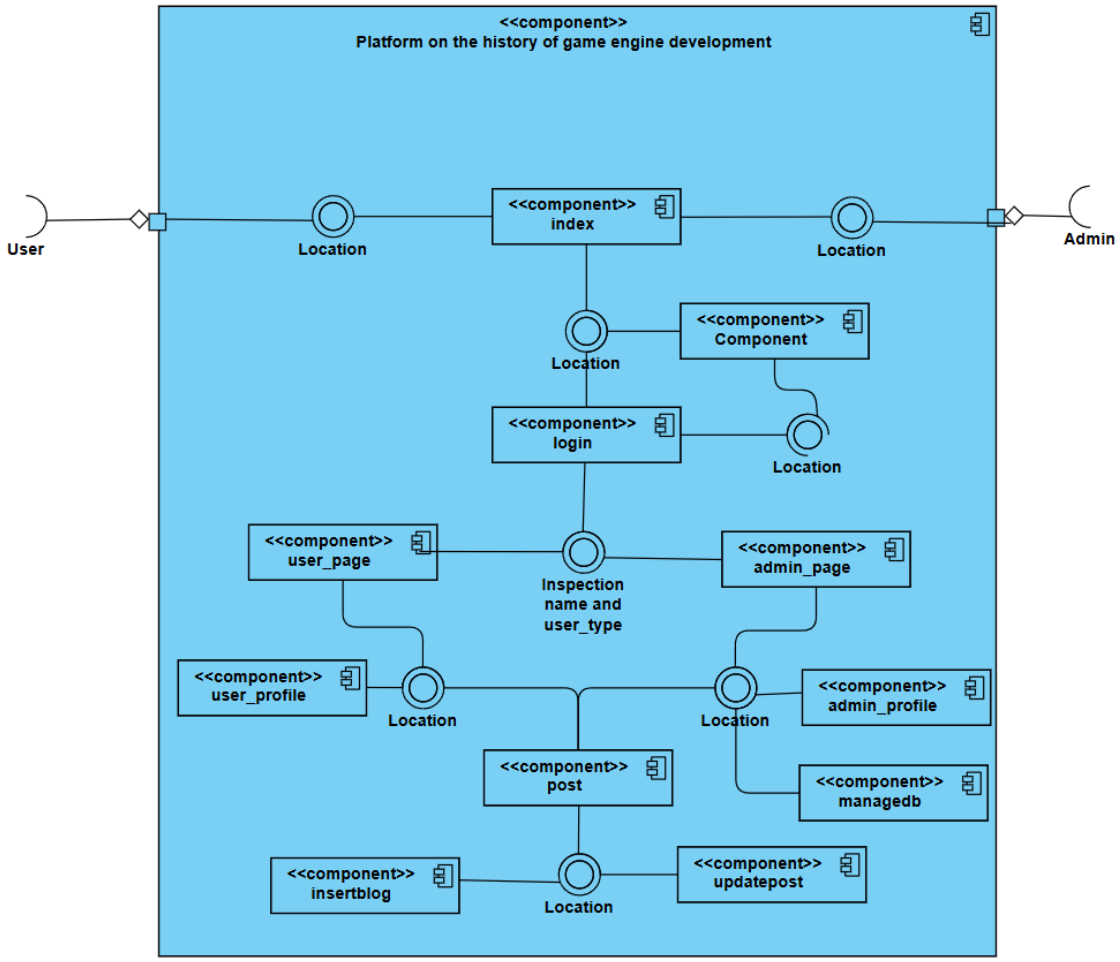


Рисунок 2.1. Діаграма компонентів (Component diagram)

Потрібно пам'ятати, що немає універсального компонента, що вирішує всі завдання. Кожен компонент має свої переваги і при проєктуванні своєї системи потрібно обирати той, що підходить саме вам. У даному випадку для розробки вебдодатку для створення статей була обрана саме діаграма компонентів.

2.2. Побудова діаграм варіантів

Процес розробки програмного забезпечення потребує системного уявлення про взаємодію користувачів із функціональністю програмного продукту. Одним з найбільш ефективних інструментаріїв на початкових етапах проєктування є діаграми варіантів використання (use case diagrams), що дають можливість візуалізувати відношення між зовнішніми суб’єктами системи

(акторами) та її внутрішніми функціями (сценаріями використання). Використання цього типу моделювання дозволяє перейти від абстрактних вимог до наочних, структурованих сценаріїв, що безпосередньо впливають на логіку та архітектуру взаємодії та розподіл доступу до ресурсів.

Основною метою побудови діаграм варіантів використання є відображення функціональних зав’язків між учасниками системи та її внутрішніми процесами, що були визначені та описані у попередньому підпункті 1.4. Зокрема, йдеться про реалізацію сценаріїв, що відповідають ролям адміністратора, користувача та зовнішньої системи. Кожна з цих ролей має власний набір прав, дій і відповідальностей, які мають бути чітко визначені у вигляді use case моделей. У свою чергу, ці сценарії відповідають раніше сформульованим функціональним вимогам та доповнюють їх візуалізованою логікою взаємодії.

Діаграми варіантів використання дозволяють:

- встановити межі системи - чітко визначити, що саме входить до складу програмного продукту, а що є зовнішнім контекстом;
- формалізувати способи взаємодії - описати, які саме дії виконує кожен актор, і як система реагує на ці дії, у якому порядку та з якими наслідками;
- визначити відповідальність та залежності між підсистемами - де кожна дія починається, на якому рівні обробляється і до яких результатів призводить;
- ідентифікувати пріоритетність функціоналу - оскільки найбільш насичені use case діаграми вказують на критично важливі модулі.

Use case моделювання виконує також важливу комунікаційну функцію. Діаграми виступають універсальним посередником між технічними та нетехнічними учасниками процесу: користувачами та адміністраторами. Вони слугують спрощеним, але точним способом представлення функціональності для обговорення, уточнення та верифікації.

Крім того, діаграми варіантів використання закладають фундамент для:

- формування логічної моделі програми (у вигляді специфікацій класів, інтерфейсів, модулів);
- розробки тесткейсів на наступних етапах;
- побудови архітектурного плану (які сервіси будуть реалізовані, які вони мають пріоритет, які є допоміжними);
- забезпечення послідовності вимог - від загального бачення системи до конкретних технічних рішень.

Тому побудова use case діаграм є критичним етапом формування концептуальної моделі системи, що базується на сформульованих функціональних вимогах та ролях користувачів. Вона дозволяє перейти від опису до структури, від побажань до конкретики, від ідеї до моделі, яка може бути втілена у програмний код.

Перед побудовою діаграм варіантів використання потрібно чітко визначити набір акторів, які беруть участь у взаємодії з інформаційною системою, та розкрити логіку їх поведінки. У межах проєктованого програмного продукту передбачається чотири основні категорії акторів: користувач, адміністратор системи та зовнішня система (інтеграційний сервіс). Кожна з цих ролей має власну модель взаємодії з платформою, набір цільових дій та рівень доступу до функціональності.

Користувач є одним із ключових учасників платформи, він виступає джерелом інформації про себе та ініціатором дій щодо створення статті. Його основні функції включають реєстрацію в системі, створення й редагування статей і керування налаштуваннями свого профілю.

Адміністратор – це користувач із підвищеними правами доступу, він забезпечує технічну та інформаційну цілісність системи. Його повноваження включають модерацію користувацького контенту, блокування або розблокування облікових статей, управління правами доступу, контроль за дотриманням правил використання платформи, ведення журналу дій користувачів та формування звітів про системну активність. Цей актор не бере

участі у стандартній бізнес-взаємодії, але забезпечує стабільність і безпечність функціонування системи на операційному рівні.

Враховуючи вищенаведені цілі та підходи до моделювання, можна перейти до формалізації сценаріїв використання, що відповідають кожному з визначених акторів. Початковим об'єктом аналізу виступає актор із роллю користувача, оскільки саме ця категорія є одним із основних генераторів взаємодії в системі. Користувач, як правило, ініціює найширший спектр дій: від створення облікового запису та створення статей та управління персональними даними.

У межах функціональної структури проєктованої системи роль здобувача охоплює як ініціативні дії, які формують нові об’єкти в базі даних, так і реактивні дії, що пов’язані з переглядом інформації, яка надходить у відповідь на запити або ініціативу з боку адміністраторів.

Побудована діаграма варіантів використання для актора «Користувача» дозволяє структурувати всі його основні поведінкові сценарії, зіставити їх із межами відповідальності системи, та виявити ключові точки взаємодії мультимедійної інтерактивної платформи з історії розробки ігрових движків з даними, логікою бізнес-процесів та іншими учасниками. Ця діаграма не лише формалізує вимоги до функціональності, а й виконує роль концептуального мосту між логікою дій користувача та структурою ендпоінтів, які будуть реалізовані в межах програмного інтерфейсу.

Діаграма побудована у відповідності до специфікації UML та демонструє всі типові дії, які можуть бути виконані кожним із основних акторів – користувачем, адміністратором та зовнішньою інтеграційною системою. Вона репрезентує структуру інтерфейсу платформи як центрального механізму обробки запитів, та відображає межі відповідальності системи на концептуальному рівні (див. рис. 2.2.)

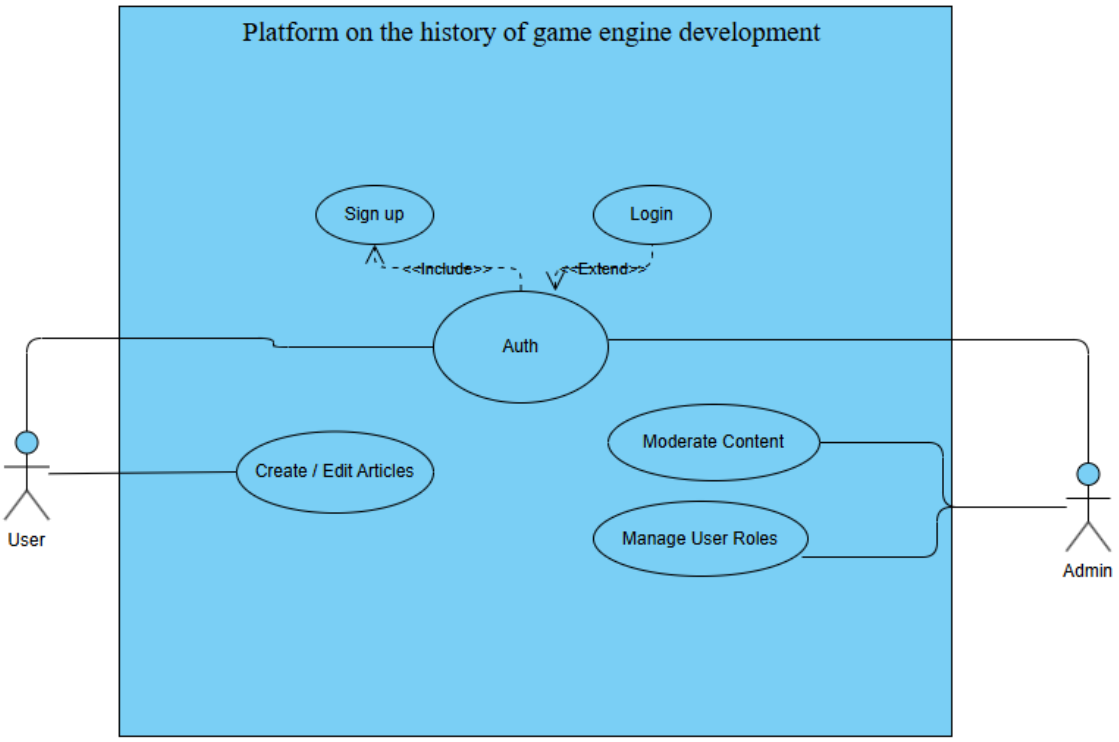


Рисунок 2.2. Use-Case діаграма

Сценарії для користувача (User)

Користувач є ініціатором більшості дій у системі, що мають на меті отримання інформації з історії розробки ігрових движків, створення статей і комунікацію з користувачами. Серед передбачених сценаріїв:

– Sign Up / Log In – створення облікового запису або вхід до системи через форму автентифікації з можливістю підтримки сторонніх сервісів.

– Create / Edit Articles – формування і оновлення статті з можливістю редагування, зміни статусу публічності, архівації.

Ці сценарії охоплюють увесь шлях користувача від входу в систему до формування результату - а саме створення статей або пошук інформації з історії розробки ігрових движків.

Сценарії для адміністратора (Administrator)

Адміністратор взаємодіє із системою переважно з метою підтримки цілісності платформи, її відповідності політикам, а також виявлення зловживань і неправдивої інформації. Сценарії включають:

– Sign Up / Log In – створення облікового запису або вхід до системи через форму автентифікації з можливістю підтримки сторонніх сервісів.

– Moderate Content – перевірка змісту статей, повідомлень на відповідність правилам (модерація до/після публікації).

– Manage User Roles – призначення, зміна або блокування ролей користувачів; управління їхніми правами доступу.

Ці функції є невід’ємною частиною системної підтримки і забезпечують безперервність, правову безпеку та прозорість платформи.

Діаграма варіантів використання на загальному рівні надає уніфіковану картину взаємодії всіх типів користувачів із системою. Вона відображає архітектурну структуру доступу до функціональності мультимедійної інтерактивної платформи з історії розробки ігрових движків, фіксує межі відповідальності кожного актора, дозволяє зрозуміти логіку розподілу сценаріїв та готує підґрунтя для наступних етапів розробки: побудови специфікацій, контролерів, інтерфейсів, моделей даних і тест кейсів.

2.3. Проєктування структури даних

Розробка програмного забезпечення у вигляді інтерактивної мультимедійної платформи з історії розробки ігрових движків передбачає необхідність формування структурованих логічних моделей даних, які забезпечують основу для зберігання, обробки та передачі інформації в межах інформаційної системи. Незважаючи на те, що фізичне проєктування бази даних у цьому контексті не є предметом реалізації, моделі даних як відображення бізнес-сутностей мають фундаментальне значення. Вони визначають логіку роботи сервісів, структуру запитів і відповідей, а також формат об'єктів, що використовуються у взаємодії між клієнтами та сервером.

Моделі даних мультимедійної інтерактивної платформи виконують роль доменних об’єктів, що описують реальні поняття предметної області: користувачів, статті, панель керування, повідомлення тощо. Кожна така сутність

містить набір атрибутів, які логічно визначають її стан, ідентифікують об’єкт у системі та дозволяють реалізовувати операції створення, оновлення, читання та видалення (CRUD). Крім того, моделі можуть включати зв’язки між сутностями, які необхідні для коректної організації інформаційної структури.

Логічні моделі є основою не лише для формування внутрішніх правил, а й для побудови мультимедійної інтерактивної платформи з історії розробки ігрових движків, а також побудови ER діаграм, що в попередньому підрозділі описують, яким чином ці сутності рухаються між користувачами, сервісами і зберігаються у відповідних сховищах. Таким чином, моделі даних є посередником між логікою процесу та архітектурною реалізацією, яка забезпечує цілісність структури на всіх рівнях системи.

У межах цього підпункту буде представлено графічне подання концептуальної моделі даних, яка включає основні сутності, їх атрибути та зв’язки, а також наведено текстовий опис кожної сутності. Моделі будуть представлені у вигляді логічних об’єктів, які відповідають вимогам функціональності програмного продукту, описаним у попередніх розділах.

Представлена ER діаграма (див. рис. 2.3.) відображає концептуальну модель основних атрибутів і сутностей цифрової платформи з історії розробки ігрових движків, структуровану у вигляді пов’язаної множини атрибутів, що реалізують доменну логіку. Всі класи представляють абстракції, які відповідають реальним елементам предметної області: користувачам, статтям, адміністраторам тощо. Діаграма також демонструє зв’язки між цими сутностями, їхню багатозначність та ієрархічну структуру.

Клас users описує загальну інформацію про користувача: name, email, password, тип користувача (user_type), пов’язані сутності user або admin.

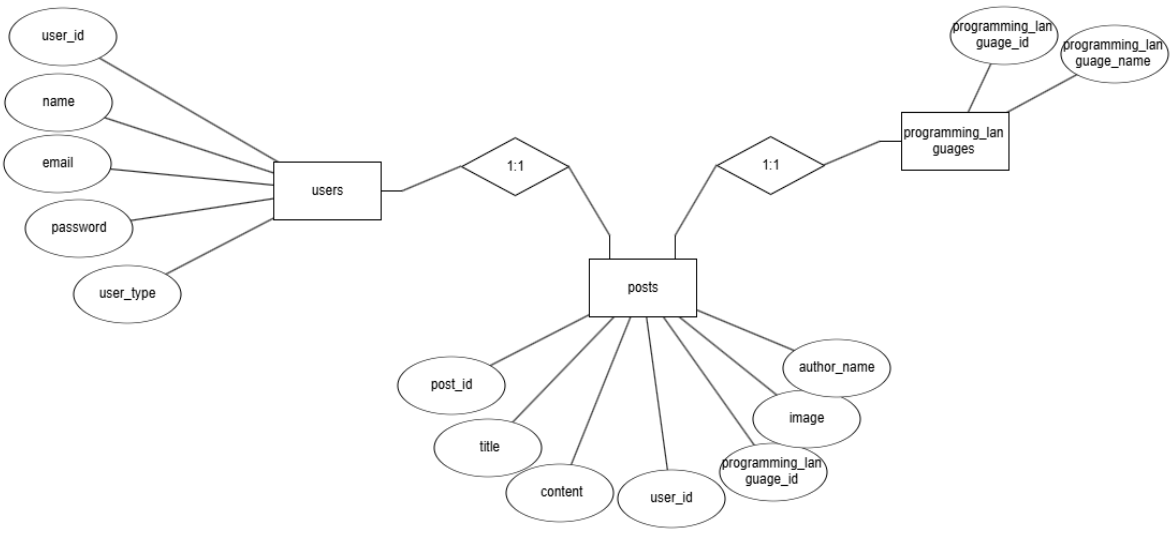


Рисунок 2.3. ER діаграма

Поле user_id є первинним ключем (primary key) унікальне ідентифікаційне поле.

Поле name псевдонім або вигадане ім'я користувача.

Поле email електронна пошта використовується для авторизації.

Поле password є важливим елементом безпеки, що використовується для захисту облікового запису користувача. Пароль складається із унікальних символів (літер, цифр та спеціальних знаків).

Поле user_type визначає роль користувача, що призначається користувачеві в обліковому записі, а також які доступи вони дають за замовчанням. Значення цієї ролі може бути або user, або admin.

Клас programming_languages представляє інформацію про мови програмування, воно містить поле: programming_language_id та programming_language_name.

Поле programming_language_id є первинним ключем (primary key) унікальне ідентифікації поле.

Поле programming_language_name назва мови програмування, яка використовується для вказівки у статтях та пості.

Клас posts представляє інформацію про створення ігрового движка і як його модифікували. В состав posts входит: post_id, title, content, user_id, programming_language_id, image і author_name.

Поле post_id є первинним ключем (primary key) унікальне ідентифікаційне поле.

Поле title містить назву поста.

Поле content містить інформацію про створення ігрового движка.

Поле user_id у таблиці posts є зовнішнім ключем, для зв'язку посту з ідентифікатором користувача.

Поле programming_language_id у таблиці posts є зовнішнім ключем, що використовується для вказівки мови програмування на якому ігровий движок був створений.

Поле image використовується для залучення уваги та розуміння теми.

Поле author_name використовується для відображення імені автора.

Зв’язок “один к одному” - при такому типові зв'язку одного запису в першій таблиці відповідає лише одному запису в іншій таблиці. В цьому випадку слід перевірити можливість розміщення всіх записів в одній таблиці. Проте у ряді випадків можна використовувати декілька простіших таблиць. Відповідність записів встановлюється по полю, яке є первинним ключем в першій таблиці, і полю, званим зовнішнім ключем іншої таблиці.

Структура моделі є багаторівневою і забезпечує повноцінну підтримку логіки платформи з історії розробки ігрових движків:

- сутності пов'язані між собою через чітко визначені зв’язки;
- передбачена можливість розширення за рахунок нових статей;
- логічні об’єкти відповідають функціональним вимогам платформи та використовуються у контролерах.

Загалом у межах підрозділу 2.3 було сформовано логічну модель даних цифрової платформи, побудовану на основі сутностей предметної області. Незважаючи на відсутність фізичної реалізації бази даних, у рамках архітектури мультимедійної інтерактивної платформи з історії розробки ігрових движків,

було обґрунтована доцільність розробки структурованих моделей сутностей, що визначають формат, зв’язки та призначення об’єктів, з якими працює програмне забезпечення.

В основі моделі знаходяться ключові бізнес-сутності: користувачі, статті, компанії, повідомлення та навички. Кожна з них описана через набір атрибутів, які відображають логіку зберігання і обробки даних, відповідно до функціональних вимог, специфікації платформи та сценаріїв взаємодії, які були описані у попередніх підрозділах. Особливу увагу було приділено визначенню зв’язків між класами, що дозволяє реалізувати логічну цілісність і підтримку складених структур – наприклад, списків статей, списків авторів тощо.

Графічна діаграма класів служить зручним інструментом для візуального представлення ієрархічних зв’язків між об’єктами системи. Вона значно полегшує правильну декомпозицію логіки обробки на рівнях контролерів і сервісів, водночас дозволяючи зберегти відповідність між структурою даних і потоками, що описуються у DFD діаграмах. Така модель також виступає своєрідною угодою між бекендом і фронтендом або зовнішніми сервісами, погоджуючи формат і зміст вхідної й вихідної інформації.

У підсумку, сформована логічна модель даних відіграла одну з ключових ролей у створенні інформаційної основи для програмного забезпечення. Вона забезпечила цілісність структури, можливість масштабування та відповідність специфіці предметної галузі, а також дала початок для подальшого архітектурного проєктування.

2.4. Проєктування поведінки

У процесі проєктування програмного забезпечення надзвичайно важливою є можливість формалізовано представити не тільки структуру системи, а й її поведінку – тобто, як саме система реагує на зовнішні події, обробляє дані і керує потоками взаємодій між користувачами та модулями. Для цього використовується набір інструментів UML (Unified Modeling Language), а

саме - поведінкові діаграми, що дозволяють описати динамічні аспекти функціонування програмного продукту.

UML є універсальною мовою візуального моделювання, яка дозволяє розробникам, і адміністраторам однаково інтерпретувати логіку системи через стандартизовані графічні позначення. Особливої уваги в межах цього підрозділу заслуговують діаграми діяльності і діаграми послідовності, що наочно демонструють алгоритм дій та взаємодію між об’єктами у часі відповідно.

Діаграми діяльності (activity diagrams) дозволяють описати потік керування процесом – від ініціації події до завершення, включаючи розгалуження, умови, паралельні дії тощо. Вони дуже добре підходять для моделювання бізнес-процесів, обробки запитів або складних сценаріїв взаємодії з користувачами. Наприклад, пошук необхідної інформації, процес створення статті або модерація контенту можуть бути описані саме у вигляді activity diagram.

Діаграми послідовності (sequence diagrams) показують, як саме об’єкти системи (користувачі, сервіси, контролери, бази даних тощо) взаємодіють між собою через повідомлення. Вони відображають не лише порядок викликів, а й часову структуру цих взаємодій. Sequence diagram ідеально підходить для опису процесів аудентифікації, створення, редагування та видалення статей.

Проєктування починається з діаграми діяльності. У системі створення, редагування та видалення статей є одним із найважливіших процесів. Цей сценарій є точкою взаємодії між двома ключовими ролями – користувачем і адміністратором – та активує взаємозв’язки між об’єктами: статтями, коментаріями та повідомленнями. Побудова діаграми діяльності для цього процесу дозволяє чітко описати алгоритм дій користувача та системні перевірки, які супроводжують цю взаємодію.

Дана діаграма діяльності є повноцінною поведінковою моделлю для всього цифрового сервісу з пошуку статей, в основі якого лежить мультимедійної інтерактивної платформи з історії розробки ігрових движків. Її структура охоплює повний життєвий цикл взаємодії між користувачем та

адміністратором. Крім того додаються технічні процеси автентифікації, управління обліковими даними, а також обробка виняткових ситуацій.

Починається діаграма зі старту сеансу, після чого ініціюється логіка перевірки: чи сесія активна і чи користувач є автентифікованим. У випадку, якщо при перевірці, автентифікація не була виконана, система пропонує користувачу пройти реєстрацію або авторизацію. У межах цього кроку можуть бути реалізовані додаткові гілки, пов’язані із відновленням пароля, підтвердженням електронної адреси або запитом на зміну пароля. Після успішного входу або реєстрації, система ідентифікує роль користувача, що й визначає подальший шлях на діаграмі.

Для користувача, який входить у систему як користувач, відкривається логіка, орієнтована на взаємодію із статтями. Основною діяльністю є створення статті, яка супроводжується параметрами фільтрації, що визначають назву ігрового двіжка, мова програмування, опис та логотип ігрового двіжка. Після створення статті можна переглядати її деталі. В іншому випадку система формує підказку про необхідність створення статті, після чого користувач переходить до заповнення профілю.

Коли стаття вже створена, користувач може редагувати записи. Цей процес доповнюється можливістю оновлення профілю або редагування вже готових статтей, що передбачає підтримку зворотного зв’язку між активністю користувача і змінною структури його облікових даних.

У випадку, коли користувач авторизований, йому відкривається інший блок діяльності, який починається зі створення профілю. Після цього активується логіка публікації статті. Система дозволяє створювати нові статті, редагувати вже опубліковані, змінювати інформацію про наповненість статті. На додаток до цього, буде реалізований блок із доступом до аналітичної інформації. Користувач може переглядати відгуки від інших користувачів, популярність статті, кількість переглядів.

Адміністраторська гілка представлена в окремому сегменті діаграми. Вона починається з перегляду контенту, що потребує модерації, а саме

повідомлень про порушення статей або активності користувачів, які викликають підозру. Адміністратор може переглядати весь контент, ухвалювати рішення про його публікацію, редагування або блокування. Окремо передбачено логіку управління користувачами: адміністратор має змогу призначати ролі, блокувати або розблоковувати облікові записи, переглядати журнали подій (логів), це дозволяє йому здійснювати аудит активності системи.

Важливою частиною діаграми є підсистема керування акаунтом, яка розташована праворуч. Вона моделює процеси, пов’язані зі зміною пароля, підтвердженням електронної пошти, активацією двофакторної автентифікації, видаленням облікового запису, а також обробкою виняткових подій – наприклад, блокування акаунта через порушення правил або невдалу спробу автентифікації. Всі ці процеси тісно інтегровані з основною логікою системи через механізми контролю доступу та безпеки та реалізовані за допомогою токенів, ролей доступу і внутрішньої політики.

Завершення сесії або вихід із системи представлені як окремий блок. Після ініціювання виходу користувачу показується підтвердження, і при отриманій згоді, виконується завершення сесії, журналювання події та повернення на стартовий екран або головну сторінку. До того ж може бути активоване відображення сповіщень про незбережені дані або надання рекомендацій для покращення взаємодії з платформою.

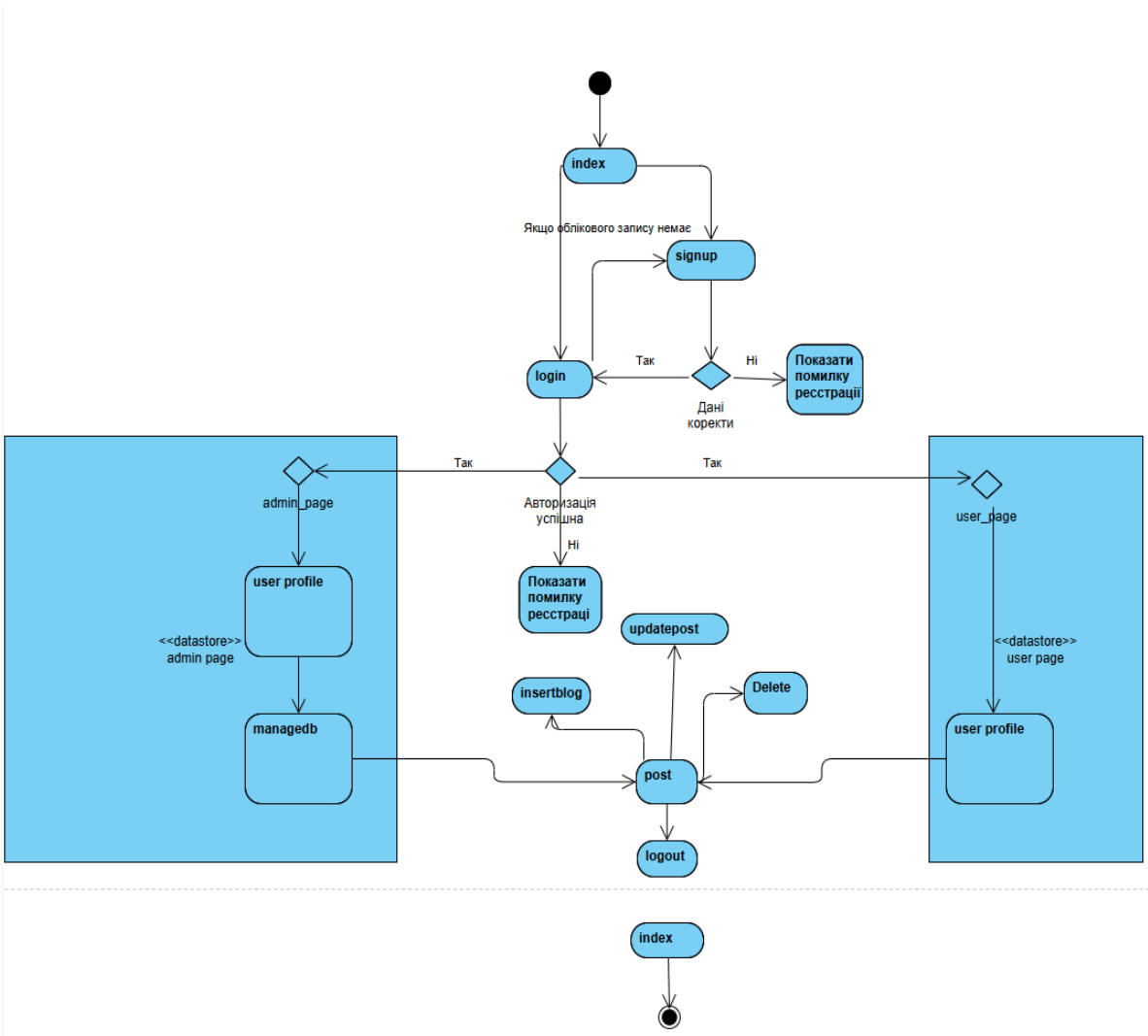


Рисунок 2.4. Діаграма діяльності (activity diagram)

Ця діаграма діяльності чітко ілюструє логіку одного з найбільш типових сценаріїв взаємодії у системі платформи. Вона не лише демонструє послідовність подій, а й дозволяє побачити важливі точки контролю, що мають бути реалізовані в login_page та admin_page сервісів, зокрема у контролерах і сервісних об'єктах платформи.

Наступною частиною буде діаграма послідовності (Sequence diagrams). Наведена на рис. 2.4 діаграма моделює повний функціональний цикл взаємодії користувачів із цифровою платформою з історії розробки ігрових движків, яку я розробляю. Вона узагальнює поведінкові процеси, які охоплюють не лише окремі дії, а і весь маршрут користувача від входу до редагування статті – із залученням різних сервісів, контролерів і баз даних. Я створив цю діаграму для

того, щоб зафіксувати не ізольовані випадки використання, і структурувати картину цілісної роботи платформи, яка реалізована в рамках мого дипломного проєкту.

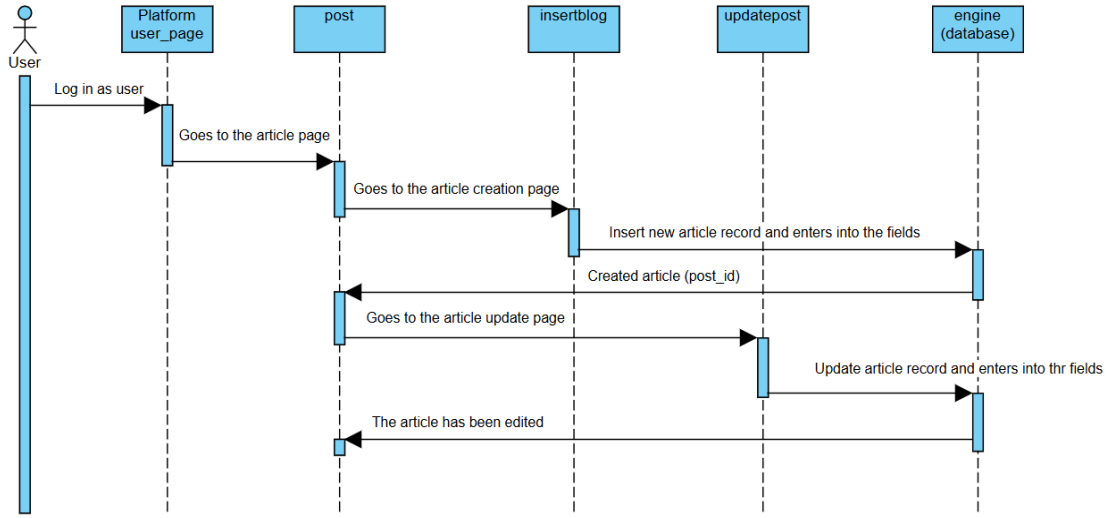


Рисунок 2.5. Діаграма послідовності (activity diagram)

Початковим тригером сценарію є ініціалізація сесії користувача, який відкриває застосунок та ініціює запит на авторизацію. Клієнтська частина, що реалізує інтерфейс користувача, передає облікові дані до бази даних (engine). Я реалізував обробку автентифікації через сервіс, що звертається до бази даних користувачів (users), звідки отримує дані для перевірки. У разі успішної авторизації, він повертається користувачу і є обов'язковим для подальших запитів до захищених маршрутів. Цей блок, як я описував раніше, є загальною точкою входу для всіх типів користувачів.

Після авторизації система переходить до гілок, які залежать від отриманої ролі. У випадку користувача, я реалізував функціональність створення статей через сторінку insertblog. Користувач надсилає форму з персональними даними, описом досвіду, навичками, після чого контролер сервісу валідує отриману інформацію і зберігає її в posts. Ідентифікатор створеної статті повертається користувачу. У діаграмі ця послідовність чітко відображає структуру контролер сервісу бази даних, що я застосовував у логиці всіх маршрутів.

Окрема частина діаграми демонструє повторне використання даних: користувач змінює раніше створене статтю, яке оновлюється через відповідні контролери. Це свідчить про підтримку CRUD парадигми, яка є фундаментом архітектури публікації, яку я обрав для свого застосунку.

Наступний блок у діаграмі ілюструє ситуацію, коли користувач створює та редагує статтю. Це окремий запит, який створює нова стаття на сторінки публікації.

2.5. Проєктування інтерфейсу

У межах даного програмного проєкту реалізовано серверну частину інформаційної системи у вигляді вебдодатку, який функціонує як проміжна ланка між базою даних та клієнтськими застосунками. В такій архітектурі сам графічний інтерфейс користувача не реалізується безпосередньо в межах цього застосунку, однак весь інтерфейсний функціонал забезпечується за рахунок ретельно спроєктованої та задокументованої взаємодії через UI.

Діаграма розгортання фокусується на фізичному аспекті виконання системи - які сервери, середовища або хости беруть участь у роботі додатку, як саме розміщено ключові компоненти, як відбувається передача даних між ними. Для веборієнтованих сервісів діаграма розгортання є основним засобом показу інфраструктури: клієнтські застосунки, сервер обробки запитів, база даних, зовнішні сервіси.

Діаграма розгортання (deployment diagram) (Рис. 2.6.) демонструє фізичну інфраструктуру, в межах якої функціонує програмний додаток. Це забезпечує високу доступність, резервування, масштабованість даних та відповідність сучасним вимогам до надійності систем.

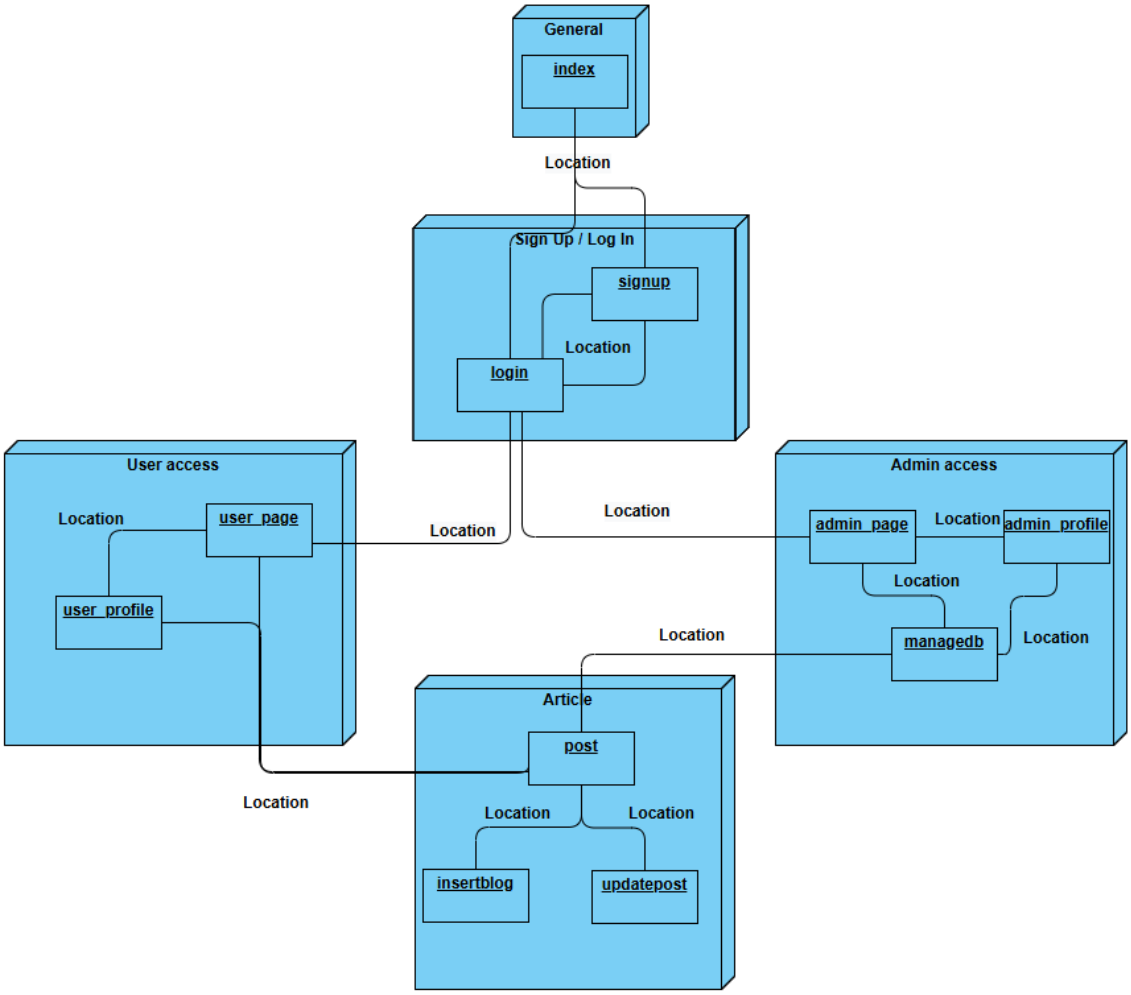


Рисунок 2.6. Діаграма розгортання

Серверна логіка веб-додатка розгортається у межах XAMPP. Тут реалізовано весь функціональний стек API: обробка запитів, авторизація, бізнес-логіка, зв’язок із базою даних. Запити надходять до платформи через HTTPS і спрямовуються до окремих сторінок, які обробляють логіку запросів, статтей, профілів, повідомлень тощо.

Усі структуровані дані зберігаються в XAMPP на основі СУБД MySQL. Це дозволяє реалізувати реплікацію, автоматичне резервне копіювання, контроль доступу а також масштабування обсягу зберігання без зупинки сервісу.

Узагальнюючи, інтерфейс користувача в даному проєкті реалізовано не як візуальний компонент, а як набір логічно структурованих сервісів, які постійно

забезпечують повноцінну взаємодію між кінцевим користувачем і внутрішніми даними системи. Це дає змогу розділити процес розробки на дві автономні частини: клієнтську (інтерфейсну) та серверну (логічну), що є ключовим принципом сучасного архітектурного проєктування на основі вебдодатку.

2.6. Висновки до розділу 2

У межах другого розділу було здійснено системний аналіз вимог до розроблюваного програмного продукту, який представляє собою серверну частину інформаційної системи з історії розробки ігрових движків з використанням платформи. Робота над розділом передбачала глибоке опрацювання функціональних і нефункціональних характеристик системи, визначення ролей користувачів, побудову моделей взаємодії та специфікацію функціональності у вигляді варіантів використання.

Побудова діаграм компонентів, активностей, діяльності, варіантів розгортання та ER model використання дозволила формалізувати логіку взаємодії між користувачами та програмним продуктом, визначити межі системи та зафіксувати ключові сценарії, які мають бути реалізовані. Кожна діаграма була супроводжена текстовим описом, який пояснює її структуру, а також визначає зв’язки між сценаріями, залежності та рівень відповідальності кожного актора.

Таким чином, в розділі 2 були побудовані діаграми, що забезпечило повноту й структурованість постановки задачі, опис вимог і поведінкову модель, що лягла в основу реалізації технічної частини системи. На основі цього розділу була побудована архітектурна логіка, визначено інтерфейси, а також сформовано основу для етапів розробки, тестування та впровадження програмного продукту.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Середовище та інструменти розробки

Програмний інструментарій для розробки вебдодатку для автоматичної агрегації статей з історії розробки ігрових движків визначався інструментарієм хостингу даного вебдодатка і аскетичність вебклієнта для досягнення MVP. А саме: вебсервер XAMPP; база даних MySQL; бек-енд PHP; HTML – утиліта для перетворення на CSS, HTML – інструментарій для браузера.

Система управління базами даних (СУБД) – це комплекс мовних і програмних засобів, що призначений для створення, ведення і сумісного використання баз даних багатьма користувачами. Зазвичай СУБД розрізняють за використовуваної моделі даних. Так, СУБД, засновані на використанні реляційної моделі даних, називають реляційними СУБД [1, 12, 19].

В якості СУБД для розробки WEB-додатку для автоматичної агрегації банківських даних було обрано MySQL. MySQL – вільна система управління реляційними базами даних.

Дана система управління базами даних (СУБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL – одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, так як має чудову підтримку з боку різноманітних мов програмування [16].

Основні приємні сторони MySQL - це багатопоточність, підтримка декількох одночасних запитів, записи фіксованої і змінної довжини, оптимізація зв'язків з приєднанням багатьох даних за один прохід, ODBC драйвер в комплекті з ісходником, гнучка система привілеїв і паролів, до 16 ключів в таблиці, кожен ключ може мати до 15 полів [16].

Також є підтримка ключових полів і спеціальних полів в операторові CREATE, підтримка чисел довжиною від 1 до 4, рядків змінної довжини і міток часу, інтерфейс з мовами C і perl. Заснована на потоках, швидка система

пам'яті, утиліта перевірки і ремонту таблиці. Всі операції роботи з рядками не звертають уваги на регістр символів в оброблюваних рядках, псевдоніми застосовні як до таблиць, так і до окремих колонок в таблиці, всі поля мають значення по замовчуванню. INSERT можна використовувати на будь-якій підмножині полів. Легкість управління таблицею, включаючи додавання і видалення ключів і полів.

Також доступний 32-бітовий драйвер ODBC для MySQL. Він дозволяє запрошувати і отримувати дані з інших джерел з підтримкою ODBC. Окрім технічних подробиць можна додати, що MYSQL який працює як на Unix, так і на платформі Windows, він дуже простий і зручний у роботі [16].

Гнучкість СУБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиці InnoDB, які підтримують транзакції на рівні окремих записів. Більш того, СУБД MySQL поставляється із спеціальним типом таблиць EXAMPLE, що демонструє принципи створення нових типів таблиць. Завдяки відкритій архітектурі і GPL-ліцензуванню, в СУБД MySQL постійно з'являються нові типи таблиць [16].

MySQL вважається хорошим рішенням для малих і середніх додатків. Вихідний код сервера компілюється на безлічі платформ. Найбільш повно можливості сервера виявляються в UNIX-системах, де є підтримка багатопоточності, що підвищує продуктивність системи в цілому [16].

Переваги MySQL [15]:

- безкоштовні засоби адміністрування від виробників;
- безкоштовні відкриті ліцензії;
- можливість використання на різних платформах (Unix, Windows, ін.);
- підтримка більшістю хостингових компаній;
- підтримується необмежена кількість користувачів, що одночасно працюють із БД.

XAMPP - це безкоштовна багатоплатформова збірка вебсервера з відкритим кодом, який децентралізований і має центральне сховище, де всі його модулі та проєкти можна легко використовувати. В основному його називають прикладним програмним забезпеченням, яке широко використовується як веб-сервер. Цей сервер підтримує різні міжплатформові операційні системи, такі як UNIX, Mac OS X та Windows, він також написаний на мовах програмування на PHP, Perl C ++ та C [21].

Він простіший у використанні і здебільшого цей сервер добре використовувати для розгортання декількох веб-додатків. Це дозволяє легко реалізувати більшість складних функцій та суттєво скорочує час розробки та використання ресурсів, надаючи існуючі функціональні можливості. Проєкти або бібліотеки можуть бути додані до проєктів як залежність або баночки та можуть використовуватися на основі функціональних можливостей та вимог програмного забезпечення [21].

Переваги [21]:

- безкоштовний та відкритий код для використання в будь-яких проєктах;
- проєкти можуть бути інтегровані з любим типом проєктів;
- більш ефективна та високоефективна у використанні функцій;
- він сумісний майже з усіма проєктами та архітектурами.

Розглядаючи різні аспекти використання PHP - скриптова мова програмування, була створена для генерації HTML-сторінок на стороні веб-сервера. PHP є однією з найпоширеніших мов, які використовуються у сфері веброзробок (разом із Java, .NET, JavaScript, Python, Ruby). PHP підтримується переважною більшістю хлстинг-провайдерів.

Простота. PHP може бути вбудована безпосередньо в html-код сторінок, які коректно обробляються PHP-інтерпретатором. PHP містить дуже велику кількість різних функцій, що позбавляє нас необхідності писати багаторядкові скрипти для виконання простого завдання. Головне для розробника - вірно вибрати функцію відповідно до конкретного завдання. До того ж, не потрібно завантажувати бібліотеки та вказувати спеціальні параметри компіляції.

PHP містить ряд готових бібліотек для роботи із популярними базами даних [21].

Безпека.

1. Засоби безпеки системного рівня. PHP можна налаштувати так, щоб вона забезпечувала максимальну свободу дій і безпеку. PHP може працювати в безпечному режимі (safe mode), що обмежує можливості застосування PHP користувачами. Наприклад: максимальний час виконання та використання пам’яті.

2. Засоби безпеки рівня програми PHP включає надійні механізми шифрування. PHP також сумісний із багатьма додатками інших розробників, що дозволяє легко інтегрувати його з захищеними технологіями електронної комерції. Вихідний код PHP не можна переглянути у браузері, так як він виконується на сервері.

Гнучкість. PHP використовується не лише у поєднанні з HTML, але й із JavaScript, WML, XML та іншими мовами програмування. PHP-код може передаватися любим браузерам і пристроям, в тому числі стільниковим телефонам, портативним комп’ютерам. PHP-код можна виконувати в режимі командного рядка.

PHP працює на різних web-серверах (Apache, Netscape Enterprise Server, Microsoft IIS, Stronghold, Zeus) і платформах (UNIX, Solaris, FreeBSD, Windows 95/98/NT/2000/XP/2003).

В рамках розробки вебдодатку виникає необхідність роботи із різними форматами файлів. Так як різні формати мають принципово різну будову, тому раціональним рішенням буде звести всі формати до одного, стандартного для нас вигляду або формату.

Це дозволить не писати засоби для роботи з усіма типами файлів, а створює можливість написати засоби для роботи з лише одним форматом і конвертери з інших форматів у обраний. Такий підхід дозволяє у майбутньому легко розширювати кількість форматів, які підгримуються, тому що для додавання підтримки нового формату необхідно лише написати конвертор. Слід

уважно опрацьовувати єдиний обраний формат, оскільки він має бути певною мірою універсальним, підтримувати збереження та відображення структур, що обробляються.

Фронтальні рамки - це інтерфейс додатків, який також дуже популярно називають інтерфейсом вебсайту, таким чином, все, що стосується кінцевого користувача, можна використовувати всередині програми. Він складається з HTML-коду, який відповідає за створення структури, таблиць стилів каскадного стилю, які використовуються для візуального форматування вебсайту, CSS-коду, який використовується для дозволу динамічних елементів, таких як слайд-шоу, розширення меню, калькуляторів тощо. Рамка саме по собі - це не що інше, як платформа, основа або фундамент, на якій присутні попередньо побудовані програмні рішення, що є вебінтерфейсом у нашому випадку. Але є багато шаблонів, які завжди доступні як частина фронтальних рамок, є багато інших процесів, які найкраще підходять для налаштування, добре підходять для поточних вимог та потреб [21].

Переваги використання CSS [8 - 9]:

- кожен вебсайт має вміст і має заздалегідь задану структуру чи макет, цей макет використовується для представлення вмісту;
- послідовний підхід - це те, що CSS пропонує своїм користувачам, вносячи невеликі зміни на ваш вебсайт, такі ж зміни відображаються і в інших частинах вебсайту;
- одна з найбезпечніших методологій кодування веброзробок, це також означає, що ваш вебсайт має більше вмісту, ніж кодів;
- CSS надає перевагу доступності, створюючи структуру та документ, який полегшує читачам екрану чітко бачити і читати вміст;
- всі вебдодатки повинні пройти критерії тестування, перед тим як вони будуть передані клієнтам;
- таблиці стилів CSS підвищують вашу адаптованість до Інтернету у всіх вищезазначених термінах та гарантують, що ваші користувачі мають однаковий вигляд.

Переваги використання HTML [21]:

- HTML легко вивчити та використовувати
- HTML безкоштовний
- HTML підтримується всіма браузерами
- HTML - це найдружніша пошукова система
- HTML простий для редагування
- HTML може легко інтегруватися з іншими мовами
- HTML легкий
- HTML є базовим для всіх мов програмування
- Відображення змін миттєво
- HTML є зручним для користувачів

HTML використовується в розробці інтерфейсу протягом багатьох років. Хоча HTML надає користувачеві всі теги, щоб додати все на вебсторінку, наприклад таблицю, зображення, гіперпосилання тощо, були і деякі недоліки, які були висвітлені в останній версії HTML, тобто HTML5, що дозволяє користувачеві вставляти графічну, мультимедійну інформацію, семантичні елементи для розробки потужних вебсайтів та послідовного вдосконалення UX.

3.2. Реалізація ключових компонентів системи

Основна мета цього підрозділу полягає у тому, щоб на основі отриманого у попередньому розділі детального проєкту виконати розробку мультимедійної інтерактивної платформи з історії розробки ігрових движків.

Розробка додатка відбувалась у редактору початкового коду Visual Studio Code на рис.3.1.

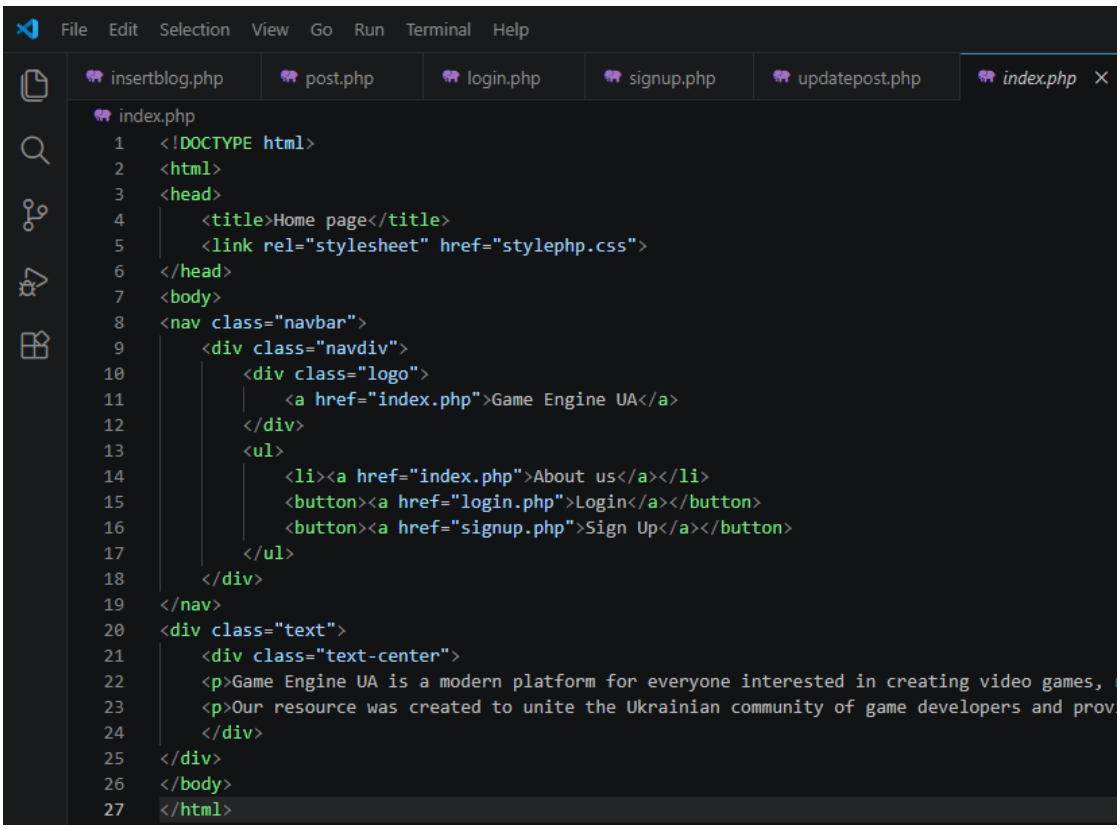


Рисунок 3.1. Фрагмент коду розробленого вебдодатку у інструментальному середовище Visual Studio Code

У межах даного програмного проєкту реалізовано серверну частину мультимедійної інтерактивної платформи з історії розробки ігрових движків, що функціонує як проміжна ланка між базою даних та клієнтськими застосунками. Сайт надає користувачу основний навігаційний елемент, розташований зазвичай у верхній частині сторінки, тому при натисканні на кнопку "About us", "Login", "Sign Up" користувача переадресовує на іншу сторінку. Головне меню - заголовок в HTML (лістининг 3.1).

Лістинг 3.1 – Головне меню (index)

```
<!DOCTYPE html>
<html>
<head>
    <title>Home page</title>
    <link rel="stylesheet" href="stylephp.css">
</head>
<body>
<nav class="navbar">
    <div class="navdiv">
        <div class="logo">
```

```
                <a href="index.php">Game Engine UA</a>
            </div>
            <ul>
                <li><a href="index.php">About us</a></li>
                <button><a href="login.php">Login</a></button>
                <button><a href="signup.php">Sign Up</a></button>
            </ul>
        </div>
    </nav>
    <div class="text">
        <div class="text-center">
        <p>Game Engine UA is a modern platform for everyone interested in creating video games, regardless of their level of experience. Here, beginners, enthusiasts, and professional developers meet to share knowledge, ideas, and experience in the field of game development.</p>
        <p>Our resource was created to unite the Ukrainian community of game developers and provide convenient access to useful information: from basic concepts to complex technical solutions. If you are just starting your journey, we will help you take the first steps. If you are already an experienced developer, here you will find new ideas, tools, and opportunities for development.</p>
        </div>
    </div>
    </body>
    </html>
```

Головна сторінка додатку містить меню на рис.3.2, яке також присутнє на кожній іншій сторінці і за допомогою якого можна швидко переміщуватись на сторінку з необхідним функціоналом.

Рисунок 3.2. Головне меню

Доступ до баз даних MySQL отримано стандартно через сценарії PHP, щоб можна було читати дані з бази та записувати дані в базу безпосередньо з вебресурсу. Підключення до сервера MySQL - використовуючи оператор mysqli_connect, було створено файл config (лістинг 3.2) [10, 18, 21].

Лістинг 3.2 – Доступ до баз даних (config)

```
<?php
$conn = mysqli_connect('localhost', 'root', '', 'engine');
```

```
if (!$conn) {
die("Connection error: " . mysqli_connect_error());
}
mysqli_set_charset($conn, "utf8mb4");
?>
```

Файл user_page та admin_page - це головна сторінка користувача після входу в систему приведено в лістинги 3.3 - 3.4 та на рис. 3.3 - 3.4. У верхній частині сторінки, розташований кнопки "Home", "Blog", "Database management (доступно лише для адміністратора)", "Edit profile", "Logout", та нікнейм користувача.

Лістинг 3.3 - Головна сторінка користувача (user_page)

```
<?php
include 'config.php';
session_start();
if(!isset($_SESSION['name']) || $_SESSION['user_type'] !== 'user'){
    header("Location: login.php");
    exit();
}
?>

<!DOCTYPE html>
<html>
<head>
    <title>User page</title>
    <link rel="stylesheet" href="stylephp.css">
</head>
<body>
<nav class="navbar">
    <div class="navdiv">
        <div class="logo">
            <a href="user_page.php">Game Engine UA</a>
        </div>
        <ul>
            <li><a href="user_page.php">Home</a></li>
            <li><a href="post.php">Blog</a></li>
            <li><a href="user_profile.php">Edit profile</a></li>
            <li><a href="logout.php">Logout</a></li>
            <li><a                href="#"><span><?php                echo
$_SESSION['name'];?></span></a></li>
        </ul>
    </div>
</nav>
<div class="text">
    <h2>Welcome, <span><?php echo $_SESSION['name'];?></span></h2>
    <p>This is an user page</p>
</div>
</body>
</html>
```

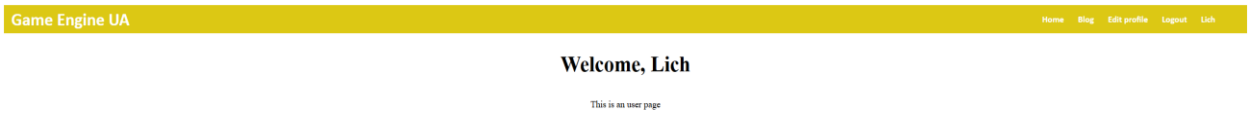


Рисунок 3.3. Головна сторінка користувача

Лістинг 3.4 - Головна сторінка адміністратора (admin_page)

```
<?php
@include 'config.php';
session_start();
if(!isset($_SESSION['name']) || $_SESSION['user_type'] !== 'admin'){
    header("Location: login.php");
    exit();
}
?>

<!DOCTYPE html>
<html>
<head>
    <title>Admin page</title>
    <link rel="stylesheet" href="stylephp.css">
</head>
<body>
<nav class="navbar">
    <div class="navdiv">
        <div class="logo">
            <a href="admin_page.php">Game Engine UA</a>
        </div>
        <ul>
            <li><a href="admin_page.php">Home</a></li>
            <li><a href="post.php">Blog</a></li>
            <li><a href="managedb.php">Database management</a></li>
            <li><a href="admin_profile.php">Edit profile</a></li>
            <li><a href="index.php">Logout</a></li>
            <li><a href="#"><span><?php echo
$_SESSION['name'];?></span></a></li>
        </ul>
    </div>
</nav>
<div class="text">
    <h2>Welcome, <span><?php echo $_SESSION['name'];?></span></h2>
    <p>This is an admin page</p>
</div>
</body>
</html>
```

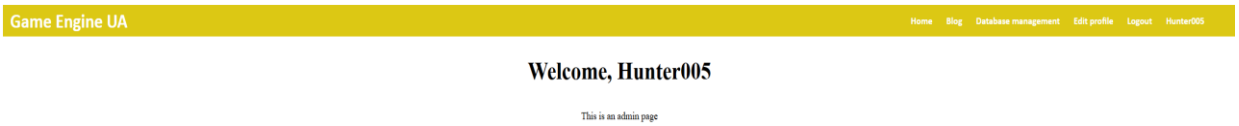


Рисунок 3.4. Головна сторінка адміністратора

Узагальнюючи, інтерфейс користувача в даному проєкті реалізовано не як візуальний компонент, а як набір логічно структурованих сервісів, які забезпечують повноцінну взаємодію між кінцевим користувачем і внутрішніми даними системи. Це дає змогу розділити процес розробки на дві автономні частини: клієнтську (інтерфейсну) та серверну (логічну), що є ключовим принципом сучасного архітектурного проєктування на основі платформи.

3.3. Реалізація бази даних та взаємодія з нею

У межах підрозділу 3.2 виконано проєктування вебдодатку, необхідної для реалізації запитів та індексування, визначених на попередніх етапах.

ORM (англ. Object-relational mapping - об'єктно-реляційна проекція) показують, як зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування. Для демонстрування ORM було створено діаграма класів приведено на рис. 3.5.

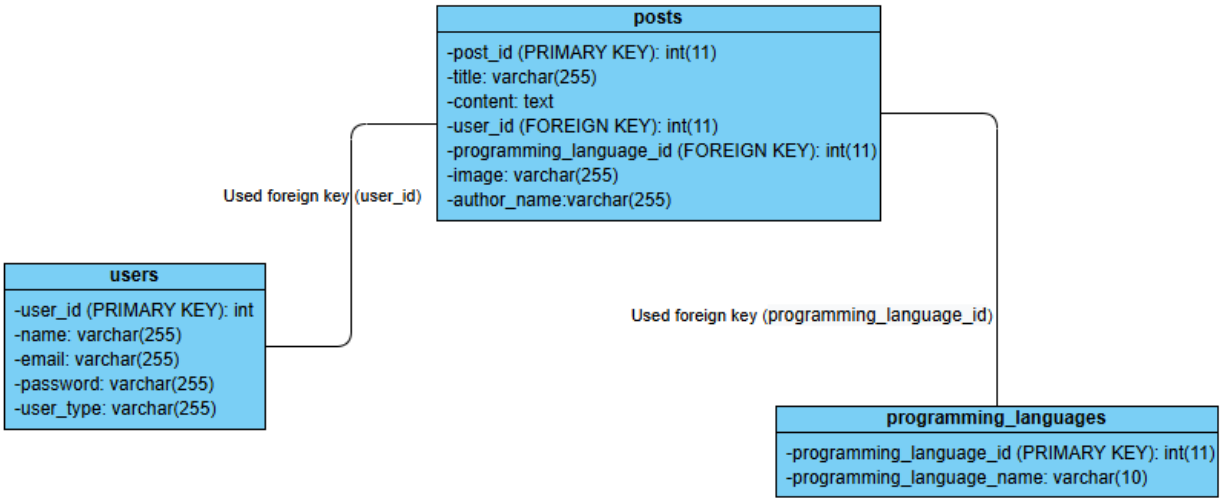


Рисунок 3.5. Діаграма класів – основні моделі

Створені акаунти можна редагувати на сторінці редагування профілю наведена у Додатку А та на рис.3.6.

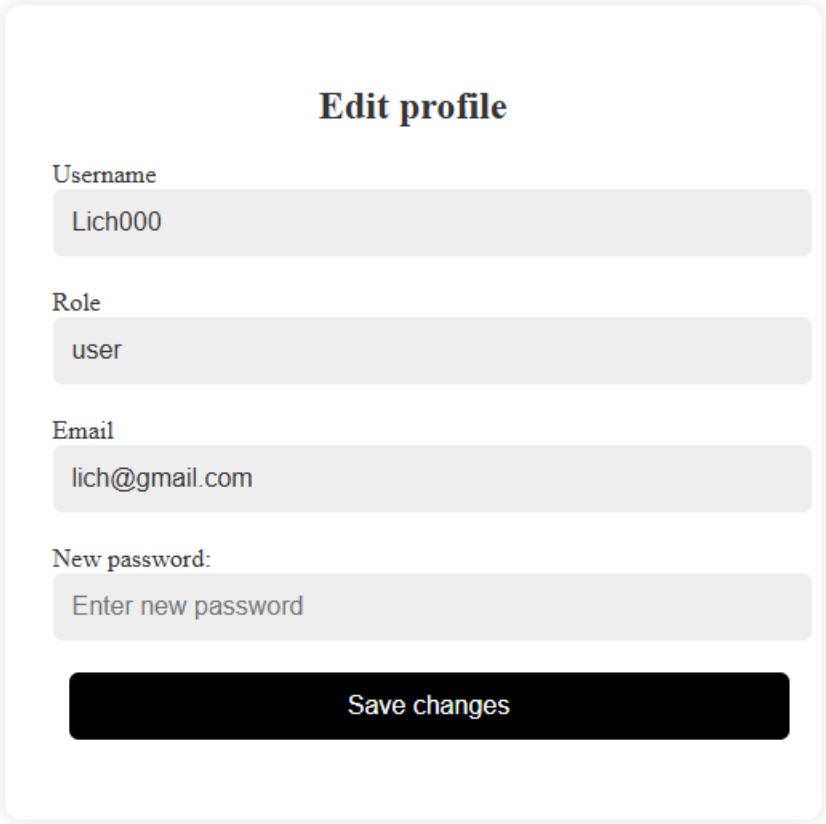


Рисунок 3.6. Сторінка редагування профілю

Для того, щоб керувати статтями були створені файли post, insertblog і updatepost, наведені у додатку А та на рис. 3.7 - 3.9.

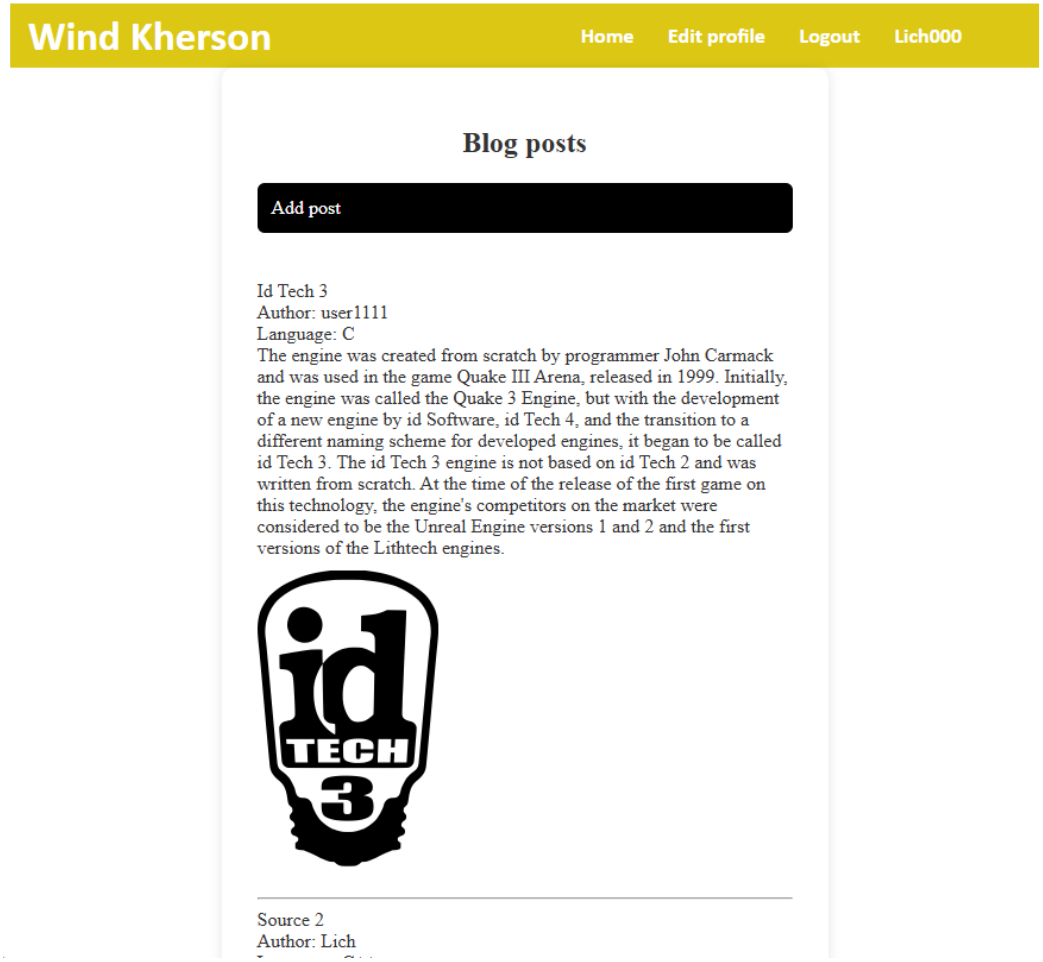


Рисунок 3.7. Сторінка з постами

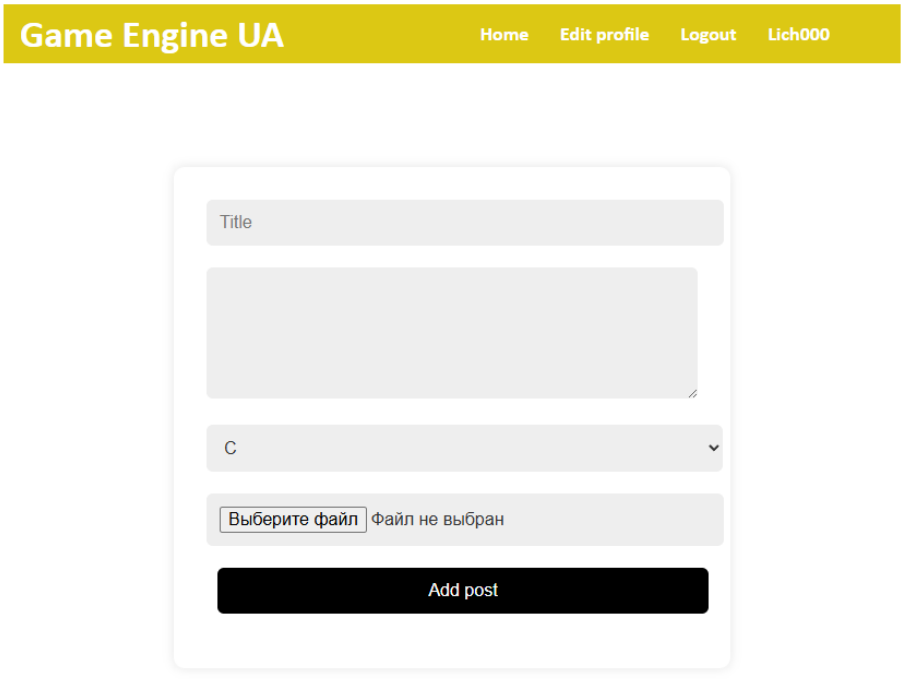


Рисунок 3.8. Сторінка публікації

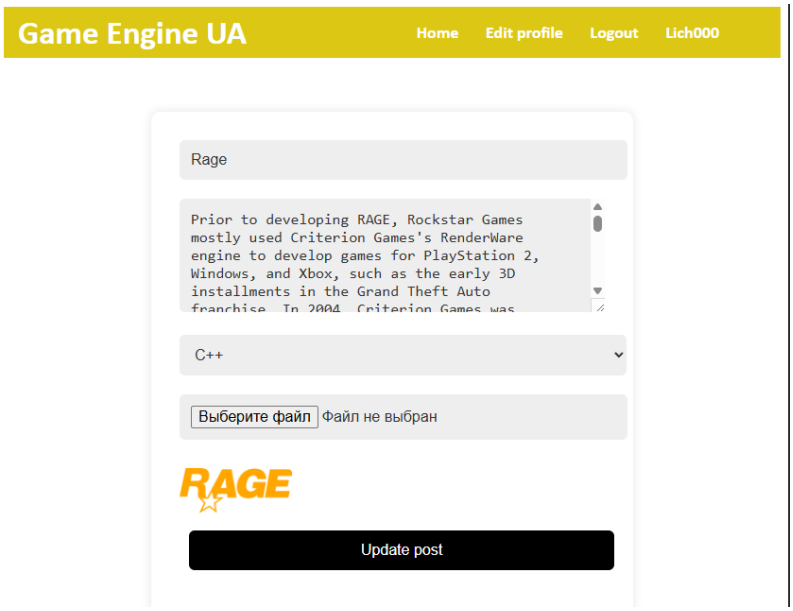


Рисунок 3.9. Сторінка редагування публікації

Файл managedb.php використовується для керування базами даними, сторінка доступна лише користувачам із роллю admin і наведена у Додатку А та на рис. 3.10.

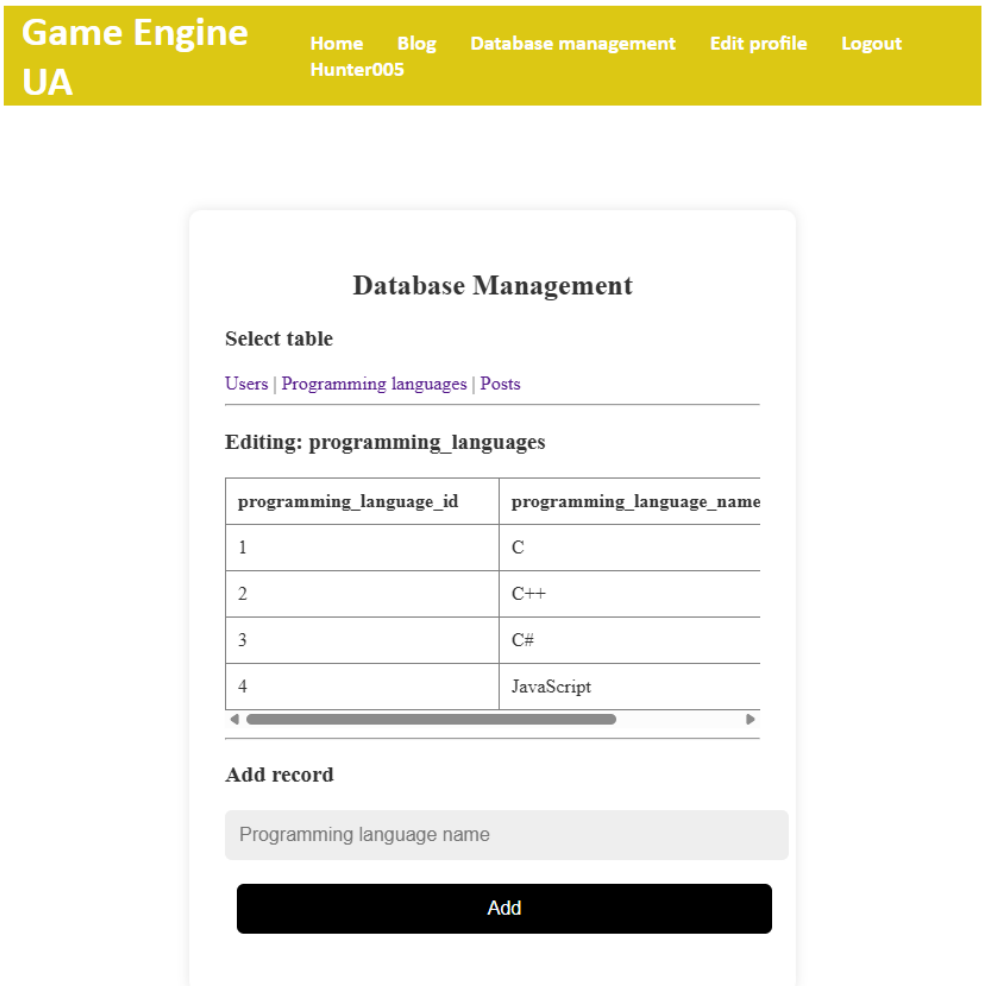


Рисунок 3.10. Сторінка панель керування базою даних

3.4. Програмування прикладного API

Формування вимог до інформаційної системи є концептуальним етапом, що закладає основу для всіх наступних фаз проєктування. Саме в межах цього процесу формалізуються очікування користувачів, відображається логіка бізнес-процесів у вигляді технічних завдань, а також визначаються ключові сценарії використання майбутнього продукту. В даному випадку йдеться про побудову API для платформи з історії розробки ігрових движків - серверного компонента, який повинен забезпечувати централізовану, стабільну та масштабовану взаємодію між усіма учасниками системи.

Для створення облікових записів користувачів та зберігання цифрових даних було створено файл signup приведено в лістинги 3.5, а також опрацьовує та відображає можливі помилки на рис.3.11.

Лістинг 3.5 - Сторінка реєстрація (signup)

```
<?php
session_start();
include 'config.php';
if(isset($_POST['submit'])){
    $name = mysqli_real_escape_string($conn, $_POST['name']);
    $email = mysqli_real_escape_string($conn, $_POST['email']);
    $password = $_POST['password'];
    $user_type = $_POST['user_type'];
    $hash = password_hash($password, PASSWORD_DEFAULT);
    $check = "SELECT * FROM users WHERE email='$email'";
    $res = mysqli_query($conn, $check);
    if(mysqli_num_rows($res) > 0){
        echo "User already exists!";
    } else {
        $sql = "INSERT INTO users(name,email,password,user_type)
                VALUES('$name','$email','$hash','$user_type')";
        mysqli_query($conn, $sql);
        header("Location: login.php");
        exit();
    }
}
?>
<!DOCTYPE html>
<html>
<head>
    <title>Sign Up</title>
    <link rel="stylesheet" href="lasu.css">
</head>
<body>
<div class="container">
    <div class="form-box-active">
        <form action="" method="post">
            <h2>Sign Up</h2>
            <input type="text" name="name" placeholder="Name" required
placeholder = "enter your name">
            <input   type="email"   name="email"   placeholder="Email"
required placeholder = "enter your email">
            <input           type="password"           name="password"
placeholder="Password" required placeholder = "enter your password">
            <select name="user_type" required>
                <option         value="user"         <?php         echo
(isset($_POST['user_type']) && $_POST['user_type'] == 'user') ? 'selected' :
''; ?>>User</option>
                <option         value="admin"         <?php         echo
(isset($_POST['user_type']) && $_POST['user_type'] == 'admin') ? 'selected'
: ''; ?>>Admin</option>
            </select>
            <input   type="submit"   name="submit"   value="Sign   up"
class="form-btn">
            <p>Already        have        an        account?        <a
href="login.php">Login</a></p>
        </form>
    </div>
</div>
</body>
```

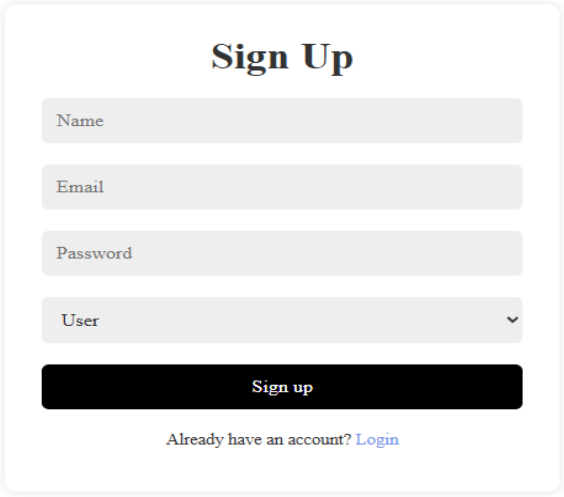


Рисунок 3.11. Сторінка реєстрація

Після створення облікового запису, користувача перекидає на сторінку авторизації (login) для входу в систему приведено в лістинги 3.6 та на рис.3.4.

Лістинг 3.6 - Сторінка авторизація (login)

```
<?php
session_start();
include 'config.php';
$error = [];
if(isset($_POST['submit'])){
    $email = mysqli_real_escape_string($conn, $_POST['email']);
    $password = $_POST['password'];
    $sql = "SELECT * FROM users WHERE email='$email'";
    $result = mysqli_query($conn, $sql);
    if(mysqli_num_rows($result) > 0){
        $user = mysqli_fetch_assoc($result);
        if(password_verify($password, $user['password'])){
            $_SESSION['name'] = $user['name'];
            $_SESSION['email'] = $user['email'];
            $_SESSION['user_type'] = $user['user_type'];
            $_SESSION['user_id'] = $user['user_id'];
            if($user['user_type'] == 'admin'){
                header("Location: admin_page.php");
            } else {
                header("Location: user_page.php");
            }
            exit();
        } else {
            $error[] = "Wrong password!";
        }
    } else {
        $error[] = "User not found!";
    }
}
?>
```

```
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
    <link rel="stylesheet" href="lasu.css">
</head>
<body>
<div class="container">
    <div class="form-box-active">
        <?php
        if(!empty($error)){
            foreach($error as $err){
                echo "<p style='color:red;'>$err</p>";
            }
        }
        ?>
        <form action="" method="post">
            <h2>Login</h2>
            <input type="email" name="email" placeholder="Email" required>
            <input type="password" name="password" placeholder="Password"
required>
            <input type="submit" name="submit" value="Login" class="form-
btn">
            <p>Don't have an account? <a href="signup.php">Sign Up</a></p>
        </form>
    </div>
</div>
</body>
</html>
```

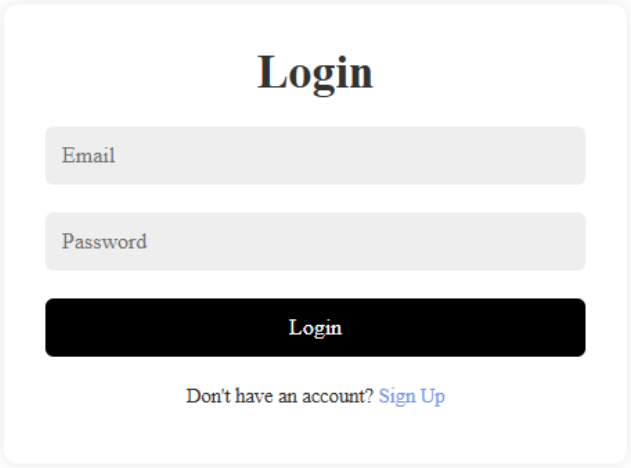


Рисунок 3.12. Сторінка авторизація

Узагальнюючи, можна стверджувати, що сформульовані вимоги до програмного продукту є результатом як аналітичного дослідження предметної області, так і синтезу інженерних практик побудови API. Вони закладають основу для створення надійної, безпечної та масштабованої серверної

платформи, яка виступатиме центральним компонентом цифрової системи пошуку статтей з історії розробки ігрових движків та забезпечить повний цикл функціонування у відповідності до сучасних стандартів.

3.5. Висновки до розділу 3

У даному розділі було обґрунтовано вибір СУБД та надано детальний опис інструментальних засобів розробки вебдодатку, наведені переваги використання саме цього програмного інструментарію для реалізації даного вебдодатку . Чітко визначена роль кожної з підсистем на всіх рівнях продукту, значною мірою спростила завдання проєктування та реалізації системи, дозволила забезпечити модифікацію продукту і його супровідність.

Розробку додатку описано із наведенням скриншотів всіх інструментальних середовищ.

Розроблений вебдодаток відповідає вимогам користувачів, володіє помірними потребами в апаратних ресурсах, заснований на переносних незалежних технологіях.

В якості перспектив розвитку наведено шляхи подальшого вдосконалення та використання додатку.

РОЗДІЛ 4. ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Модульне тестування

Починаємо тестування веб-ресурсу з перевірки кросплатформеності у сучасних популярних браузерах: Google Chrome, Mozilla Firefox та Internet Explorer. Отримані результати вказують на відсутність значних відмінностей в відображенні структури вебдодатку та дизайнерського оформлення (рис.4.1).

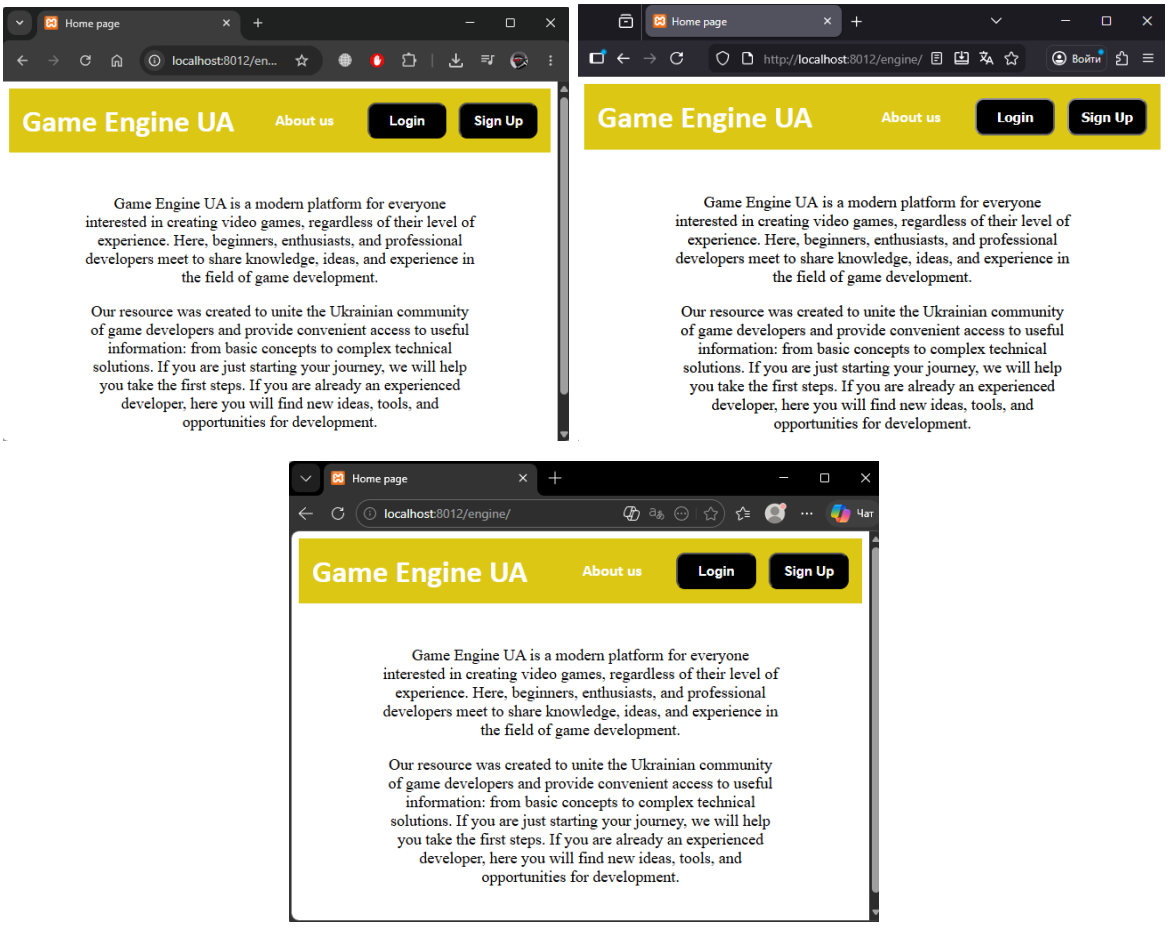


Рисунок 4.1. Перевірка кросплатформеності розробленого вебдодатку у сучасних популярних браузерах

При зміні розміру екрану констатуємо, що всі елементи змінюють місце розташування та зберігають логічну структурну побудову на рис.4.2.

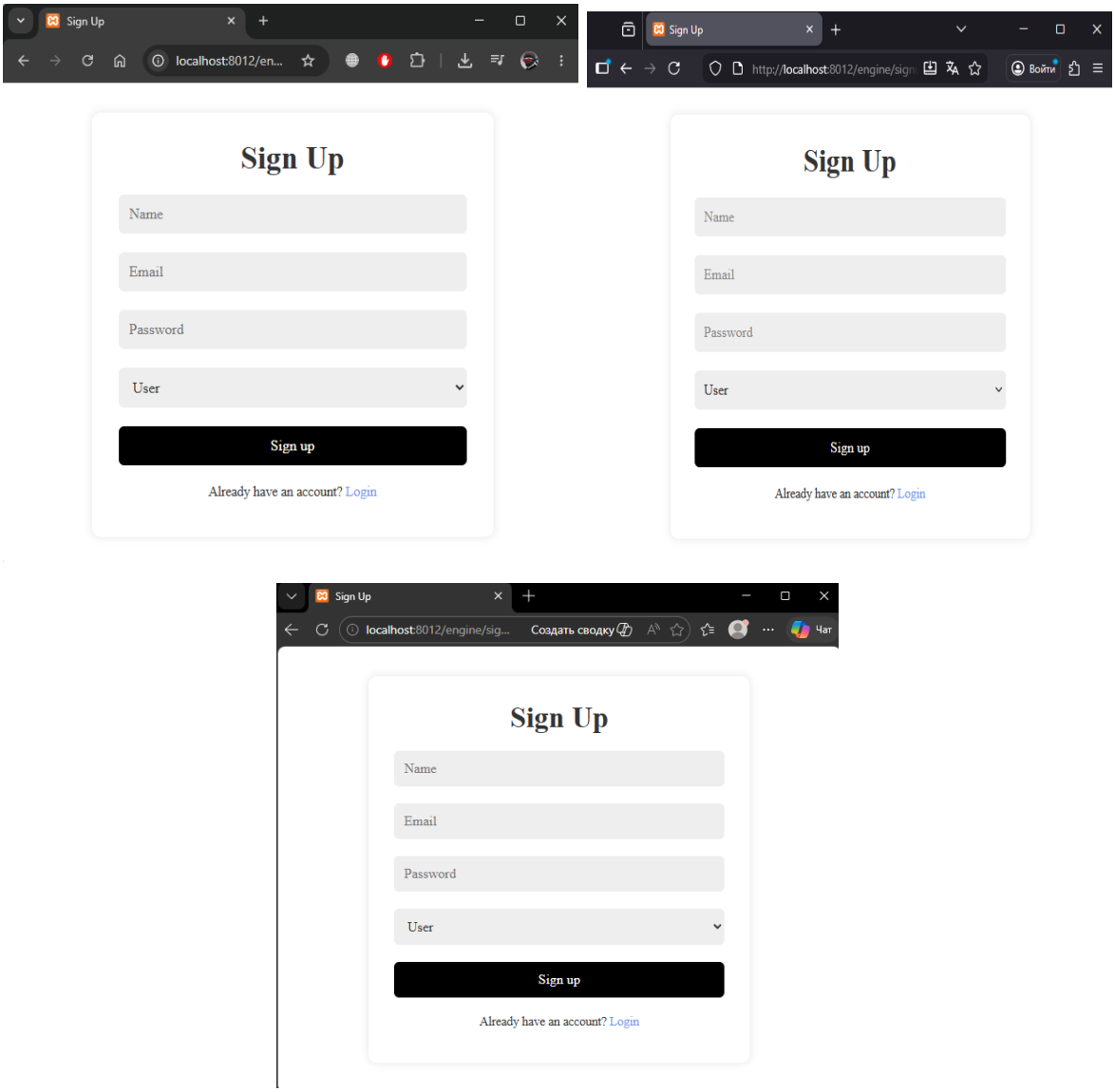


Рисунок.4.2. Перегляд роботи із вебдодатком при різних розмірах вікна

Завершальним кроком є перевірка взаємозв'язків між сторінками веб ресурсу, а саме наявність всіх посилань та відсутність переходів. Всі посилання та переходи працюють адекватно та правильно, приведено на рис.4.3.

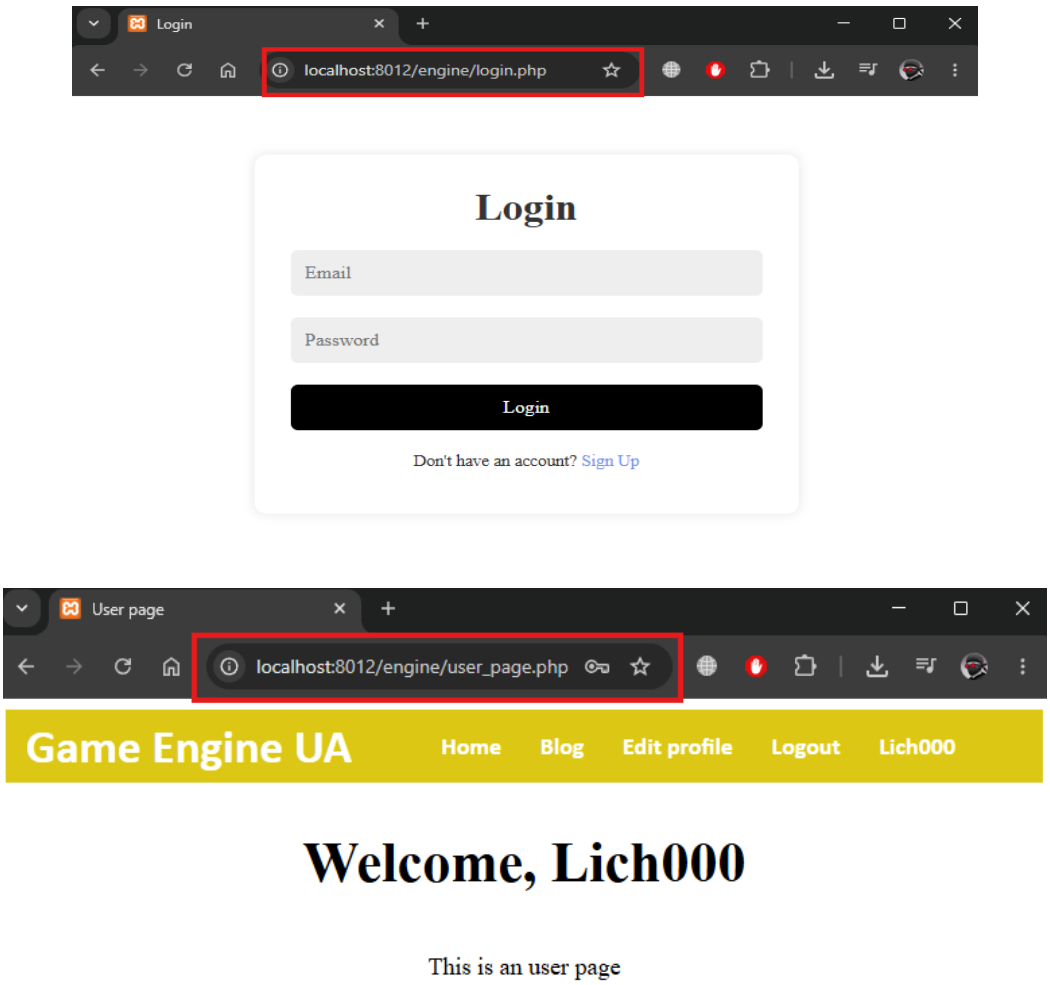


Рисунок 4.3. Перевірка працездатності та правильності посилань

При тестуванні даного вебдодатку не було виявлено порушень в його роботі, тому розроблений платформи з історії розробки ігрових движків відповідає усім технічним вимогам.

4.2. Інтеграційне та UI-тестування

Модульне тестування є базовим і найбільш ізольованим рівнем тестування програмного забезпечення. На цьому етапі перевіряється функціональність окремих програмних одиниць - переважно методів та сервісів - без залучення зовнішніх залежностей або інфраструктури, таких як база даних, контролери чи мережеві виклики. У розробленому проєкті, який реалізує

багаторівневу архітектуру платформи, модульне тестування відіграє вирішальну роль у забезпеченні надійності прикладної логіки.

Кожен модульний тест містить попередню підготовку тестового оточення, вхідні дані, виклик методу, що перевіряється, і верифікацію результату.

Реалізація тестування відбувалася з використання програму Postman для тестування з API, частина результатів тестування. Усі приклади запитів і відповіді, включно зі скріншотами Postman, наведено у додатку Є.

Кожен тест реалізує один із ключових сценаріїв взаємодії з платформи. Наприклад, запит signup перевіряє поведінку системи при успішній реєстрації користувача: викликається HTTP POST-запит на http://localhost/engine/signup.php, і перевіряється, що у відповіді роль користувача та ідентифікатор (рис 4.4).

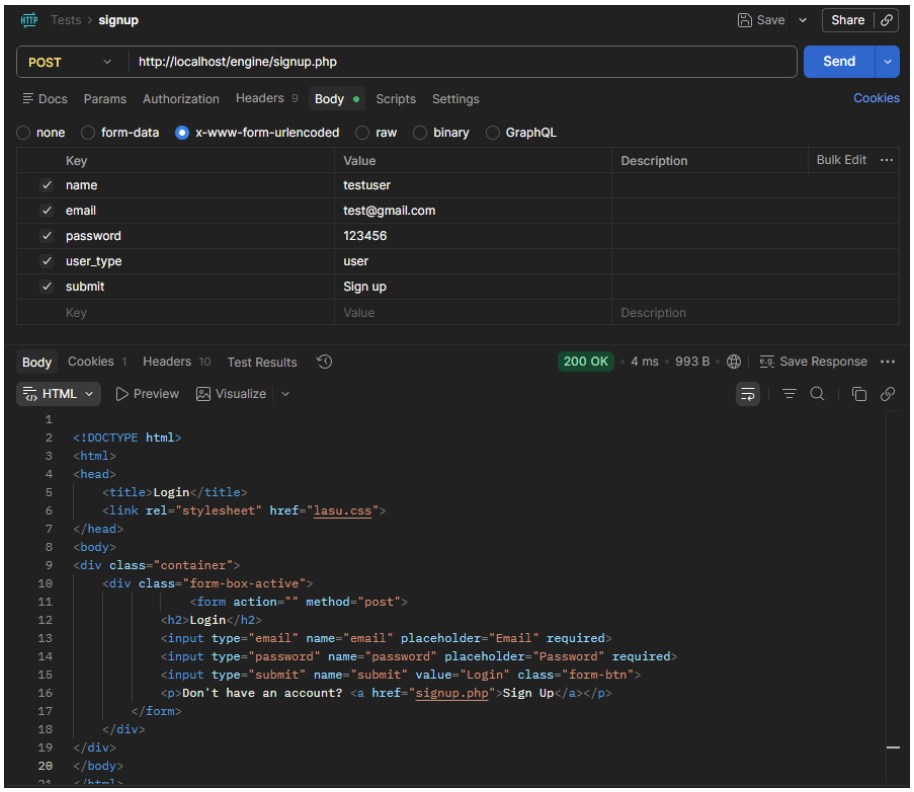


Рисунок 4.4. Результат тестування

Окремо тестуються й сценарії авторизації: наприклад, запит login перевіряє, що при правильних облікових даних система повертає супутні

атрибути, тоді як HTTP POST-запит на http://localhost:8012/engine/login.php гарантує, що при неправильному паролі користувачу буде повідомлено «Wrong password!» (рис 4.5).

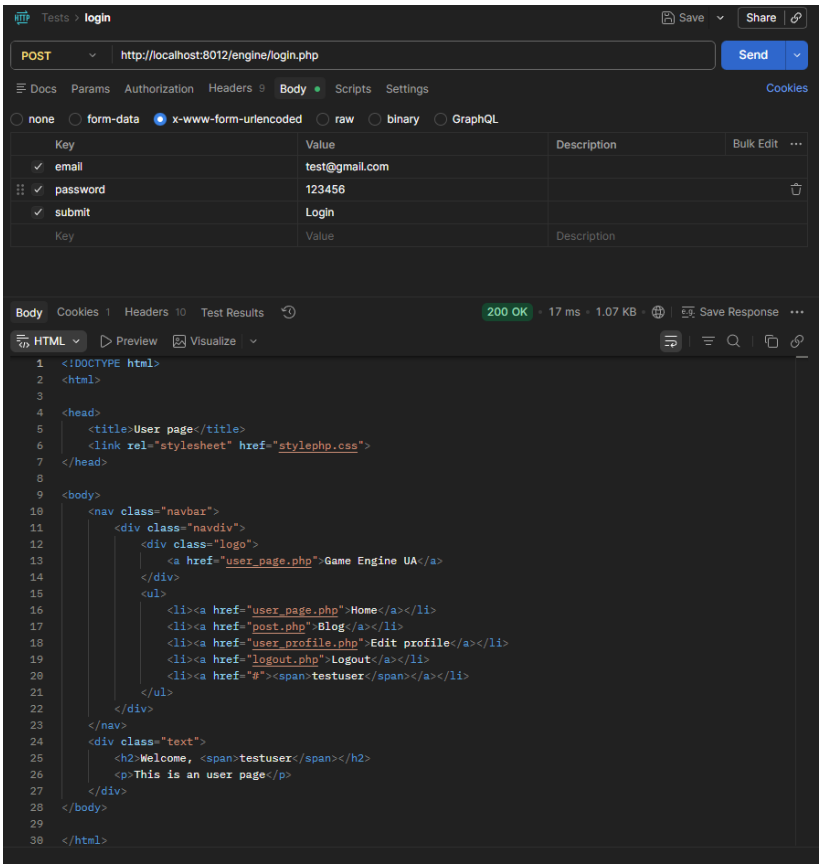


Рисунок 4.5. Результат тестування

Сценарії редагування профілю: запит user_profile перевіряє поведінку системи при успішній реєстрації користувача: викликається HTTP POST-запит на http://localhost:8012/engine/user_profile.php, що при неправильному вводі даних користувачу буде повідомлено «Error updating profile.» (рис 4.6).

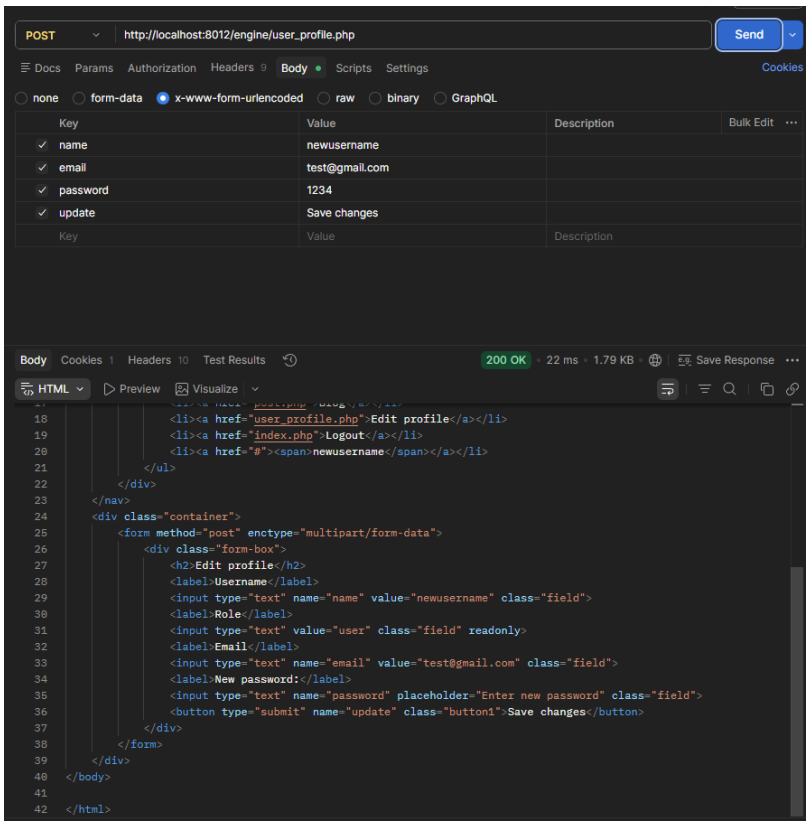


Рисунок 4.6. Результат тестування

Сценарії створення статті: запит insertblog перевіряє створення статті, система повертає супутні атрибути, тоді як HTTP POST-запит на http://localhost:8012/engine/insertblog.php, що перевіряє публікацію та ідентифікатор статті (рис 4.7).

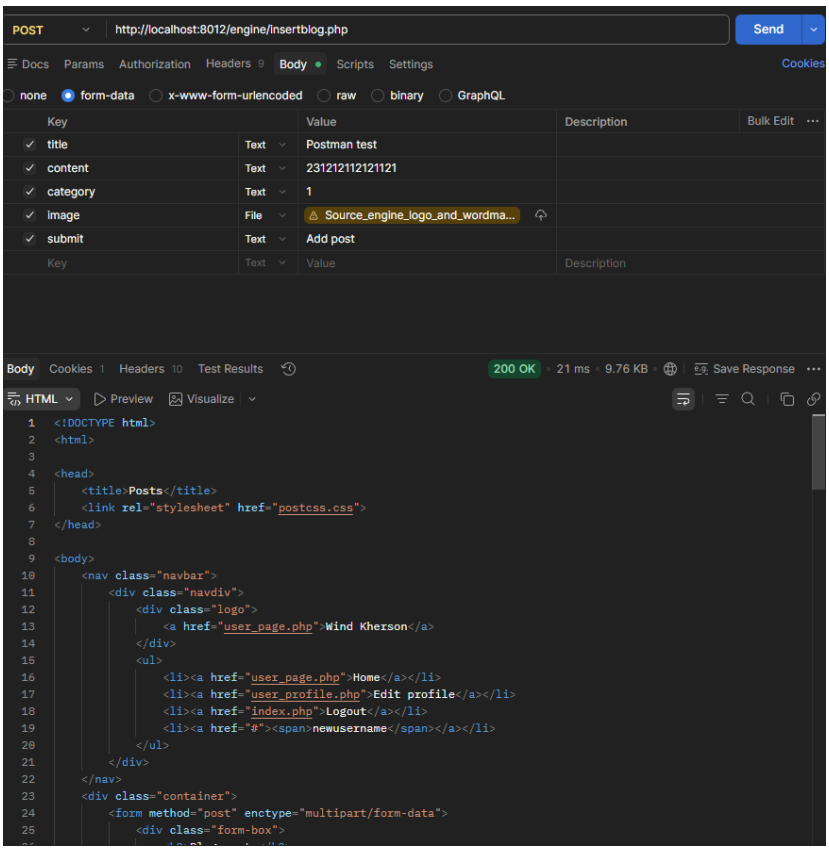


Рисунок 4.7. Результат тестування

Сценарії редагування статті: запит updatepost перевіряє редагування статті, система повертає супутні атрибути, тоді як HTTP POST-запит на http://localhost:8012/engine/updatepost.php?post_id=16, що перевіряється ідентифікатор статті (рис 4.8).

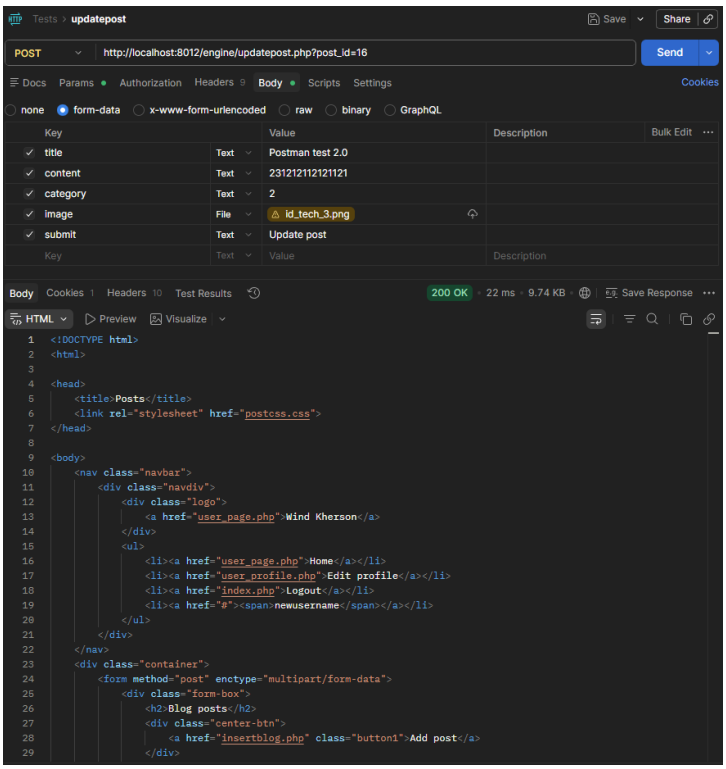


Рисунок 4.8. Результат тестування

Наведена таблиця демонструє, що пріоритетність тестів розподілена відповідно до ризиків, які ті чи інші функції становлять для стабільності роботи системи. Зокрема, найбільший пріоритет мають ті сценарії, які пов’язані з критично важливими створення, редагування або видалення статті, що найчастіше виконуються в реальному середовищі, для цього була створена таблиця, що наведена у Додатку А.

За результатами тестування було підтверджено, що всі основні компоненти додатку успішно функціонують як окремо, так і у взаємодії між собою. Усі тестові запити повертали коректні статус-коди, відповідні повідомлення та структури відповіді. Особливу увагу було приділено перевірці некоректних сценаріїв - таких як відсутність ідентифікатора, спроба отримати неіснуючу статтю, помилки в параметрах фільтрації.

Проведене інтеграційне та системне тестування підтвердило відповідність реалізованого програмного продукту функціональним вимогам, стабільність при обробці запитів, коректну обробку помилок та готовність до подальшого розгортання в продуктивному середовищі.

Усі виявлені недоліки були проаналізовані, а відповідні зміни внесені у код із забезпеченням повторної перевірки. У підсумку, проведене модульне тестування дало змогу локалізувати та усунути потенційні помилки до того, як логіка інтегрується в повний сценарій взаємодії з користувачем через платформу.

Тестові приклади, що підтверджують описаний процес, наведено у відповідному додатку до пояснювальної записки. Кожен із них супроводжується прикладом коду, вхідними параметрами, очікуваним результатом та зазначенням класу, що тестується.

Таким чином, модульне тестування стало надійною основою для контролю якості коду на початковому рівні і забезпечило упевненість у правильності реалізації окремих функцій до їх інтеграції у загальну архітектуру системи.

4.3. Аналіз результатів тестування

Тестування проведене у підрозділі 4.2, підтвердило відповідність реалізованого програмного продукту функціональним вимогам. Якість програмного забезпечення безпосередньо визначається не лише правильністю реалованої логіки, а й повнотою перевірки усіх сценаріїв його використання. Саме тому важливим етапом завершення життєвого циклу розробки є тестування, яке має за мету підтвердити відповідність програмного продукту визначеним вимогам, специфікаціям та очікуванням кінцевих користувачів.

У процесі створення тест-плану для розробленого вебдодатку враховувались вимоги до основних бізнес-процесів системи, проєктні рішення щодо архітектури, а також ключові сценарії використання. Архітектура застосунку побудована за принципами багаторівневої моделі із чітким поділом логіки на контролери, репозиторії та структури. Це дозволило виокремити функціональні частини для перевірки й побудувати логічну структуру тестування.

Основна увага була зосереджена на перевірці стабільності та правильності роботи HTTP-ендпоінтів. Усі перевірки реалізовувались із врахуванням як позитивних, так і негативних сценаріїв. Позитивні перевірки охоплюють успішні запити із передбачуваними результатами, а негативні - ситуації, пов’язані з некоректними даними, помилками доступу або відсутністю ресурсів. Значне місце в тест-плані займають також граничні випадки, які дозволяють оцінити реакцію системи на мінімальні, максимальні або нестандартні параметри. Крім того, система проходить перевірку на дотримання правил авторизації - а саме, для кожного типу користувача визначено рівень доступу до функціоналу та перевіряється його дотримання у режимі тестування.

До тест-плану включено двадцять тестових випадків, кожен з яких має унікальний ідентифікатор, стислий опис, зазначення модуля системи, очікуваний результат, тип тестування (функціональне, безпекове, негативне, граничне) та пріоритет. Структура тестів побудована так, щоб охопити всі основні категорії операцій: створення, отримання, оновлення, видалення, автентифікація, контроль доступу, валідація полів, конфлікти й винятки.

Під час тестування особливу увагу було зосереджено на таких аспектах:

– взаємодія користувача з обліковим записом (реєстрація, автентифікація, доступ до захищених маршрутів);

– створення, редагування, видалення та перегляд статтей;

– перевірка інтеграції між контролером повідомлень.

Функціональні перевірки зосереджені на стабільності взаємодії з контролерами, правильності відповідей і відповідності структури даних. Негативні сценарії дозволяють перевірити, як система реагує на порушення вимог - відсутність ресурсів, недопустимі параметри, дублювання записів. Безпекові тести перевіряють поведінку системи за наявності неправильних ідентифікаторів, спроб доступу до захищених маршрутів або виконання дій без відповідних прав. Граничні випадки включають сценарії з екстремальними розмірами вхідних даних, порожніми або нульовими значеннями.

Наведена таблиця демонструє, що пріоритетність тестів розподілена відповідно до ризиків, які ті чи інші функції становлять для стабільності роботи системи. Зокрема, найбільший пріоритет мають ті сценарії, які пов’язані з критично важливими компонентами, або ті, що найчастіше виконуються в реальному середовищі.

У процесі системного тестування платформи запускалося у повному режимі, з підключеною реальною базою даних MySQL та конфігурацією безпеки. Всі основні API-ендпоінти перевірялись на відповідність функціональним сценаріям, заздалегідь описаним у проєктних специфікаціях. Тестування проводилось як автоматизовано, так і вручну - з використанням Postman, що дозволили виконати запити в умовах, наближених до поведінки клієнтського застосунку.

Середовище, у якому проводилось системне тестування, включало:

– ОС: Windows 11 (x64);
– PHP8.2.12;
– Середовище запуску: Visual Studio Code;
– База даних: MySQL;
– Локальний веб-сервер: XAMPP;
– Засоби перевірки: Postman.

Тестування здійснювалося поетапно: спочатку на рівні окремих сервісів (unit-тести), далі - на рівні контролерів (інтеграційні тести), після чого - через ручну перевірку ключових запитів за допомогою інструмента Postman. Всі результати тестування були зафіксовані у вигляді таблиці, з позначенням фактичного результату та статусу виконання.

Таким чином, наведений тест-план є підтвердженням того, що розроблений додаток відповідає проєктним очікуванням, і слугує методичною основою для перевірки стабільності, відповідності вимогам і готовності до розгортання.

4.4. Інструкція користувача

Розроблений програмний додаток реалізовано як серверну частину інформаційної системи з пошуку статтей з історії розробки ігрових движків. Система базується на архітектурі платформи, розробленій з використанням PHP та XAMPP. Зважаючи на це, процес встановлення не потребує безпосередньої інсталяції на пристрій кінцевого користувача, а передбачає розгортання серверного застосунку у вигляді самостійного сервісу, до якого надалі можна підключати фронтенд-застосунки, мобільні клієнти, сторонні сервіси або адміністративні інструменти.

При виборі інфраструктурного середовища для розгортання платформи були розглянуті декілька хмарних платформ, зокрема Google Cloud Platform (GCP) та Amazon Web Services (AWS). GCP вирізняється глибокою інтеграцією з ML/BigQuery та зручними інструментами CI/CD через Cloud Build. Однак AWS було обрано як найбільш зріле рішення з розвиненою екосистемою сервісів для обслуговування вебдодатків, масштабування контейнеризованих сервісів через ECS/EKS. Крім того, AWS надає безкоштовний рівень для тестового середовища.

Запуск додатку можливий на будь-якому пристрої, де встановлено необхідне середовище виконання PHP. Для локальної інсталяції необхідно виконати такі кроки:

– Встановити PHP - проєкт реалізовано на версії 8.2.12, тому коректна робота гарантована саме з цією версією.

– Встановити MySQL - база даних для зберігання структурованих даних. Після встановлення потрібно створити базу даних із назвою engine, користувача і задати пароль;

– Запустити програму у середовищі розробки (наприклад, Visual Studio Code);

Для зручності розгортання у віддаленому середовищі, реалізовано підтримку XAMPP. Це дозволяє уникнути залежності від ОС, забезпечити кращу портативність і прискорити розгортання.

Підготовка середовища

На сервері необхідно:

– Встановити Apache;

– Встановити MySQL;

– Відкрити порти 80, 8012 (для Apache) і 3306 (для MySQL).

Після встановлення додатку було проведено серію перевірок повного циклу взаємодії в тестовому середовищі. Зокрема було:

– зареєстровано користувача з роллю user;

– створено профіль адміністратор;

– опубліковано тестову статтю;

У процесі експлуатації система продемонструвала стабільність, передбачуваність поведінки, правильну обробку помилок, а також наявність захисту маршрутів згідно з авторизаційною логікою. Особливо цінним є розділення прав доступу за ролями, що підтвердило успішну реалізацію безпечної архітектури.

Таким чином, запропонований програмний додаток успішно розгортається як у локальному, так і у серверному середовищі, не вимагає складних дій від кінцевого користувача, забезпечує повноцінну інтерактивну документацію, підтримку авторизації та надійне оброблення даних. Його експлуатаційні характеристики дозволяють впроваджувати систему як основу для клієнтського вебзастосунку або мобільного рішення без доопрацювання серверної логіки.

4.5. Впровадження та розгортання

Вимоги до клієнтської машини та серверу мінімальні, необхідна лише можливість працювати у веббраузері, пристрої вводу та виведення інформації.

Вебдодаток розміщений на GitHub за адресою: https://github.com/InquisitorHunter/enginePHP (рис.4.9).

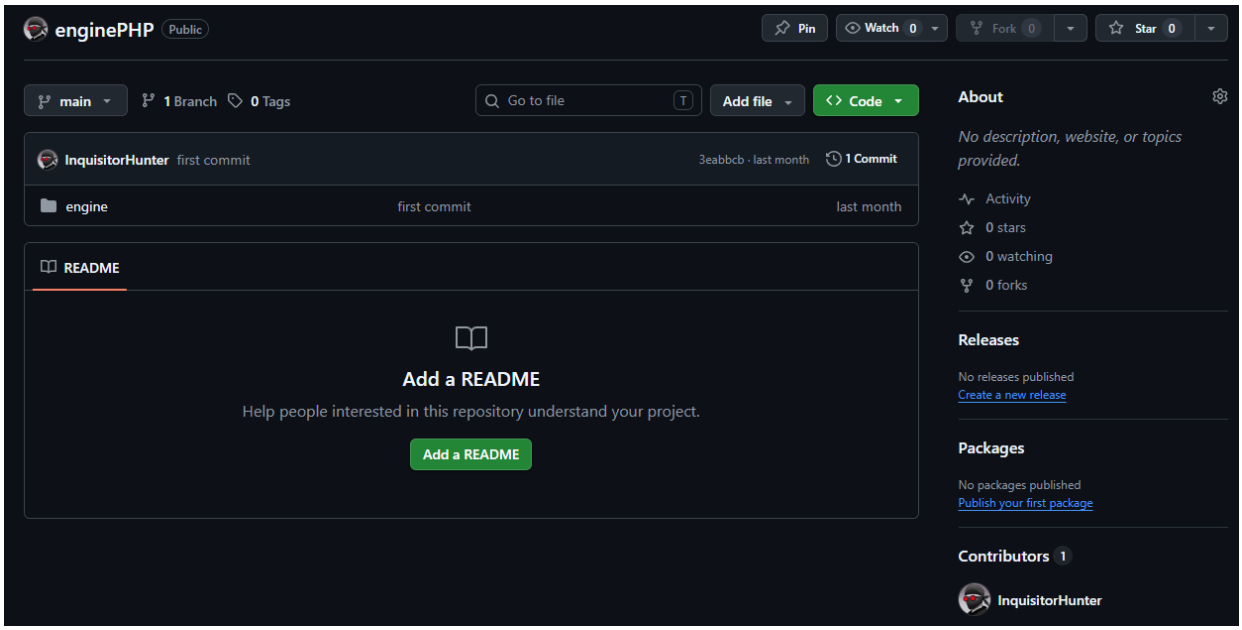


Рисунок 4.9. Проєкт на GitHub

Далі для завантаження створеного проєкту можна використовувати вбудований файловий менеджер з консолі хостингу (рис.4.10).

Рис.унок 4.10. Використання вбудованого файлового менеджеру з консолі хостингу

4.6. Висновки до розділу 4

У межах четвертого розділу було проведено повноцінний цикл тестування, наведено переваги використання саме цього програмного інструментарію для реалізації даного вебдодатку.

У підрозділі 4.1 було проведено модульне тестування. Ефективність тестування значною мірою залежить від наявності чітко структурованого тест-плану - документа, який регламентує процес перевірки програмного забезпечення.

У підрозділі 4.2 вебдодаток пройшов ряд тестових випробувань на предмет виявлення помилок у проєктуванні та реалізації. Проведено інтеграційне тестування ключових сервісів за допомогою Postman. Перевірка підтвердила відповідність вимогам функціональності, надійності, мобільності та зручністю використання.

У підрозділі 4.3 охоплено процес повного тестування програмного продукту. Побудовано тест-план, що включає понад 10 тест-кейсів різних типів. Зафіксовано низку виявлених і усунутих помилок, що підвищило надійність і якість реалізації.

У підрозділах 4.4 і 4.5 було розроблено документацію, орієнтовану на кінцевого користувача. Надано керівництво користувача з прикладами використання платформи, а також інструкцію з встановлення й розгортання системи як у локальному, так і в серверному середовищі за допомогою XAMPP. У результаті експлуатаційного тестування підтверджено працездатність ключових функцій: реєстрації, авторизації, створення, редагування, видалення, статей, та адміністративний контроль.

В якості перспектив розвитку наведено шляхи подальшого вдосконалення та використання додатку.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи бакалавра було створено платформа з історії розробки ігрових движків

В епоху розвитку інформаційних технологій та збільшення об’єму інформаційних джерел стає актуальним питання створення статті.

У процесі виконання роботи було проаналізовано предметну область, сформульовано мету, завдання та обґрунтовано необхідність автоматизації ключових функцій створення, редагування та видалення статтей. Побудовано функціональні та інформаційні моделі, визначено вимоги до архітектури, розроблено діаграми варіантів використання, послідовності, компонентів, розгортань, класів, моделей даних, потоків та сценаріїв.

Під час виконання проєкту було проведено аналіз методів та алгоритмів для проєктування та розробки вебдодатку із досліджуваної предметної області, який доводить правильність їх вибору, а саме: програмний інструментарій визначався інструментарієм хостингу даного вебдодатку та аскетичністю вебклієнта. А саме: вебсервер XAMPP; база даних MySQL; бек-енд PHP; CSS, HTML - інструментарій для браузера. Розроблений вебдодаток для платформи з історії розробки ігрових движків даних дозволяє створювати, редагувати та видалити статті, а на шляху до цих даних, що дозволяє автоматично оновлювати дані у разі зміни на оригінальних сайтах. Вебдодаток відповідає вимогам користувачів, володіє помірними потребами в апаратних ресурсах.

Особливу увагу було приділено питанням безпеки, тестованості та зручності у використанні. Здійснено модульне, інтеграційне та системне тестування з використаним програмою Postman. Розроблено та задокументовано тест-план, виявлено та усунуто типові помилки, підтверджено стабільну роботу системи при коректному навантаженні. Результати тестової експлуатації показали, що додаток відповідає технічним вимогам і забезпечує надійне виконання поставлених функцій.

Розгортання додатку було реалізовано як у локальному середовищі розробника, так і на віддаленому сервері за допомогою XAMPP. Надано повну інструкцію з встановлення, керівництво користувача, а також ілюстрації роботи з API, що підтверджують його доступність для широкого кола користувачів.

У випадку розростання сервісу, можна добавити форум для обміну інформації між користувачами додатку. В якості перспектив розвитку у дослідженні наведено шляхи подальшого вдосконалення та використання додатку.

Повний лизтінг розробленного вебдодатку приведено в GitHub URL: https://github.com/InquisitorHunter/enginePHP

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бази даних і СУБД. URL: https://timeweb.com/ru/community/articles/bazydannyh-i-subd-1

2. Вікіпедія CSS - URL: https://uk.wikipedia.org/wiki/CSS

3. Вікіпедія HTML - URL: https://uk.wikipedia.org/wiki/HTML

4. Вікіпедія PHP - URL: https://uk.wikipedia.org/wiki/PHP

5. Вікіпедія Visual Studio Code - URL: https://uk.wikipedia.org/wiki/Visual_Studio_Code

6. Вікіпедія XAMPP - URL: https://uk.wikipedia.org/wiki/XAMPP

7. Вікіпедія Діаграма компонентів URL: https://uk.wikipedia.org/wiki/%D0%94%D1%96%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%B0_%D0%BA%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D1%96%D0%B2

8. Додонов О.Г., Ланде Д.В., Путятін В.Г., Жигало В.В. Архітктура системи моніторингу, адаптивного агрегування та узагальнення інформації. Реєстрація, зберігання і обробка даних. 2013, т.15, №4. С. 32-40.

9. Коваленко І. В. Програмні інтерфейси (API): проєктування та реалізація [Електронний ресурс]. - Київ, 2022. - Режим доступу: https://kneu.edu.ua/publications/kovalenko-2022

10. Користування цифровими додатками в Україні, 2019. URL:https://newsroom.mastercard.com/eu/files/2019/06/Digitalization_infographics_UA.jpg

11. HR Tech 2023: глобальні тренди [Електронний ресурс]. - Режим доступу: https://hrm.ua/trends

12. Що таке база даних?. URL: http://apeps.kpi.ua/shco-take-basa-danykh

13. CSS-CascadingStyleSheets URL: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS3

14. McKinsey. Digital Transformation of Recruitment [Електронний ресурс]. - 2023. - Режим доступу: https://www.mckinsey.com

15. Microsoft. CI/CD Pipelines with GitHub Actions [Електронний ресурс]. – Режим доступу: https://docs.github.com/en/actions

16. MySQL Documentation URL: https://dev.mysql.com/doc/

17. Redis Documentation [Електронний ресурс]. Режим доступу: https://redis.io/docs

18. UML User Case Diagrams : Guidelines URL: https://msdn.microsoft.com/en-us/library/dd409432.aspx

19. Vasile C. Choosing the best code editor as a web developer in 2019 URL: https://designrevision.com/best-code-editor/

20. Wiegers K., Beatty J. Software Requirements. – 3rd ed. – Redmond : Microsoft Press, 2013. – 560 с.

21. Wodehouse C. A Beginner's Guide to Back-End Development URL: https://www.upwork.com/hiring/development/abeginners-guide-to-back-end-development

ДОДАТОК А

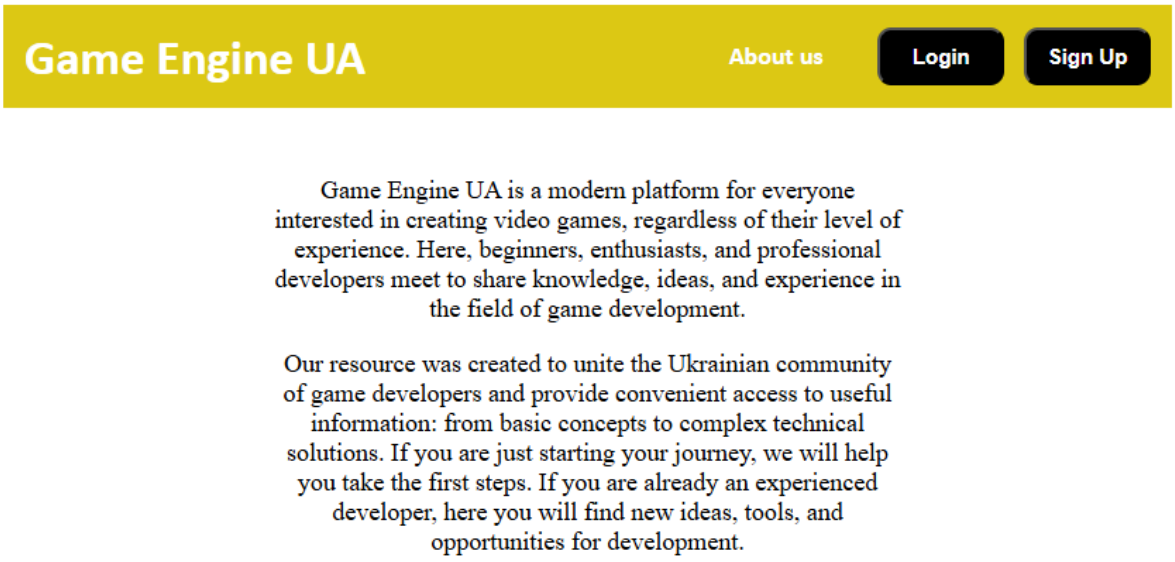
ТАБЛИЦЯ СПЕЦИФІКАЦІЙ REST API

URL	Метод	Опис	Ролі доступу	Тіло / параметри	Відповідь
http://localhost:8012/engine/	GET	Головне меню	Public	-	200 OK
http://localhost/engine/signup.php	POST	Сторінка реєстрація	Public	name, email, password, user_type, submit	200 OK
http://localhost:8012/engine/login.php	POST	Сторінка авторизація	Public	email, password, submit	200 OK
http://localhost:8012/engine/user_page.php	GET	Головна сторінка користувача	user	-	200 OK
http://localhost:8012/engine/user_profile.php	POST	Сторінка редагування профілю користувача	user	name, email, password, update	200 OK
http://localhost:8012/engine/post.php	GET	Сторінка з постами	Public	-	200 OK
http://localhost:8012/engine/insertblog.php	POST	Сторінка публікації	Public	Title, content, category, image, submit	200 OK
http://localhost:8012/engine/admin_page.php	GET	Головна сторінка адміністратора	admin	-	200 OK
http://localhost:8012/engine/ admin_profile.php	POST	Сторінка редагування профілю адмінівстратора	admin	name, email, password, update	200 OK
http://localhost/engine/managedb.php	GET	Сторінка панель керування базою даних	admin	-	200 OK

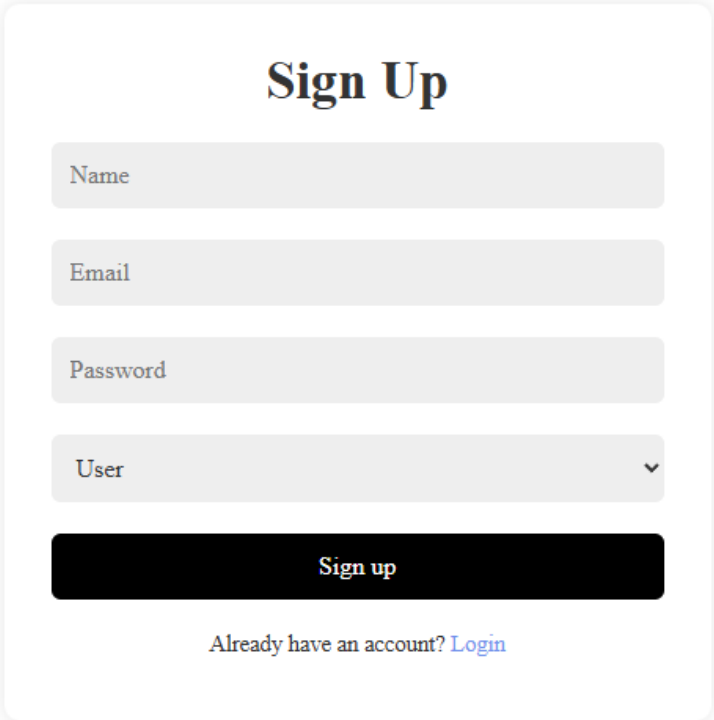
ДОДАТОК Б

ІНТЕРФЕЙС ПЛАТФОРМИ

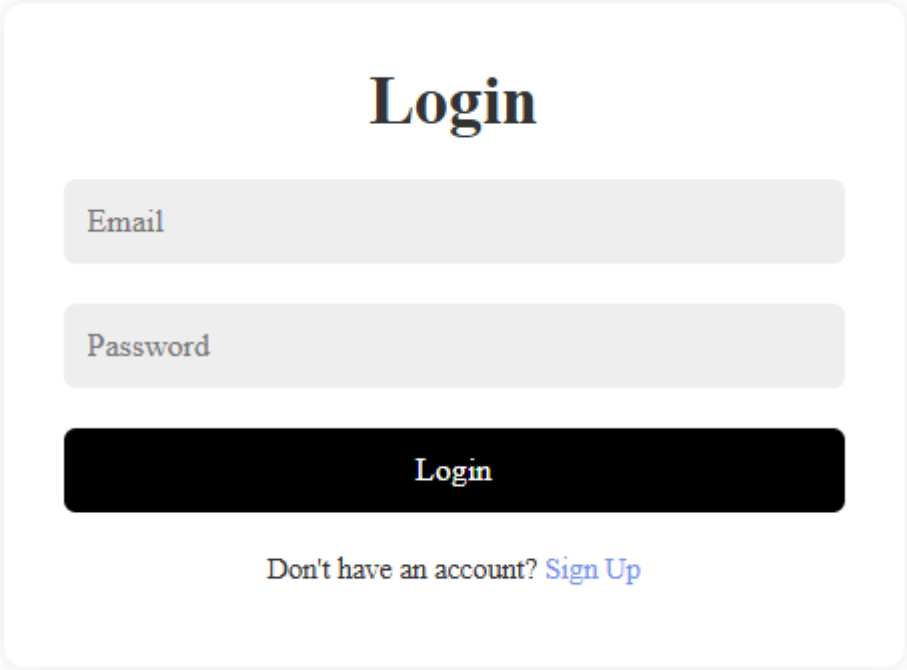
Б.1. Головне меню



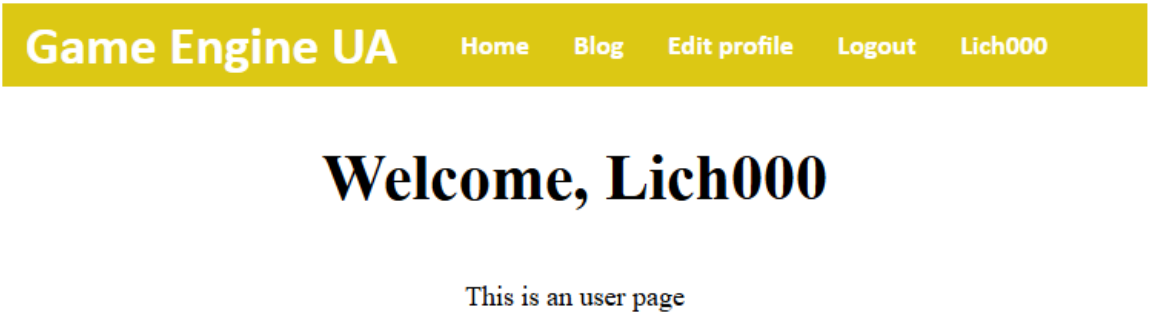
Б.2. Сторінка реєстрація



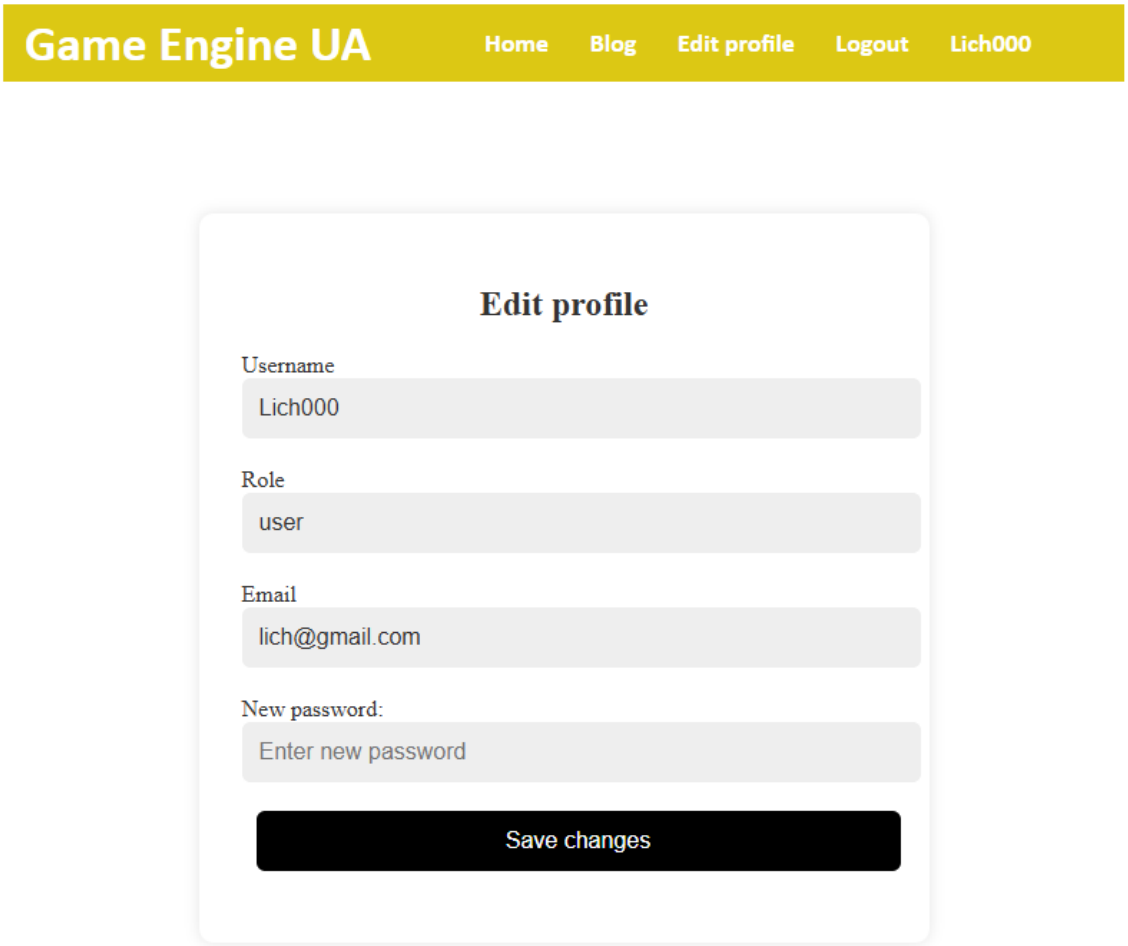
Б.3. Сторінка авторизація



Б.4. Головна сторінка користувача

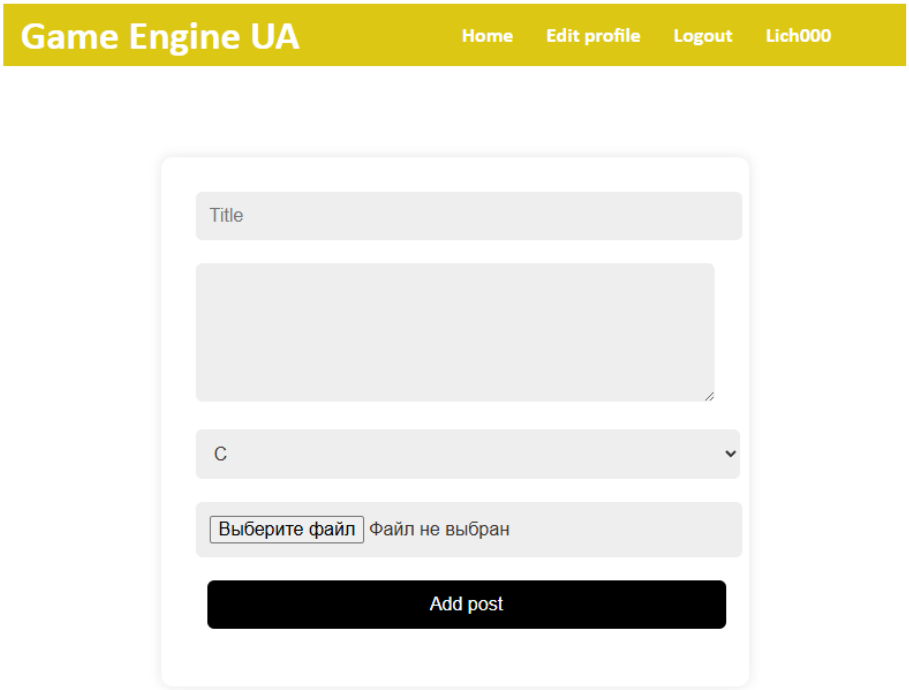


Б.5. Сторінка редагування профілю користувача

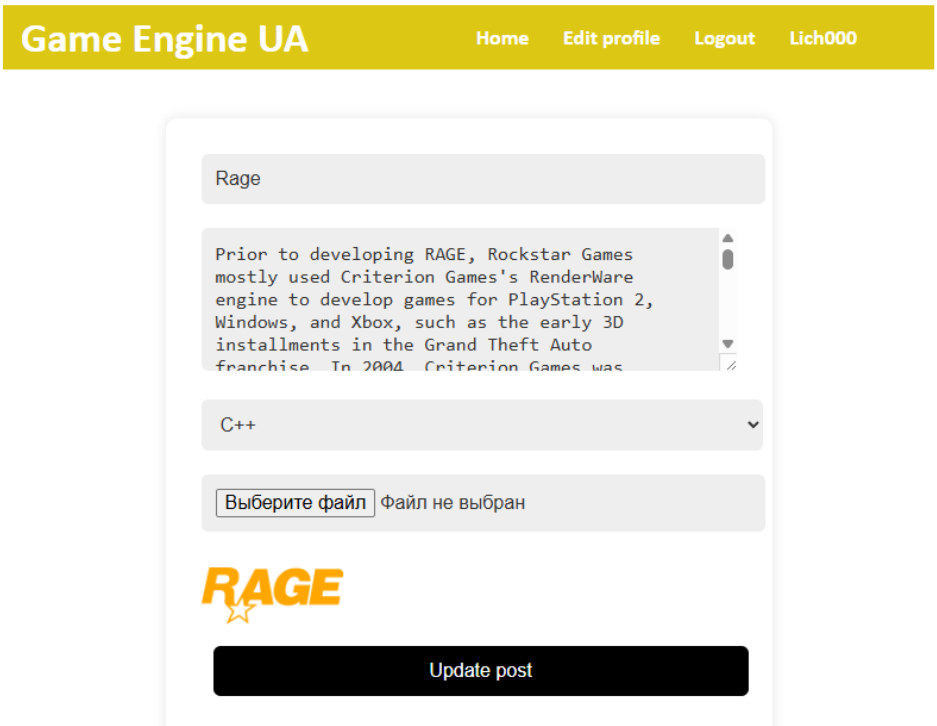


Б.6. Сторінка з постами

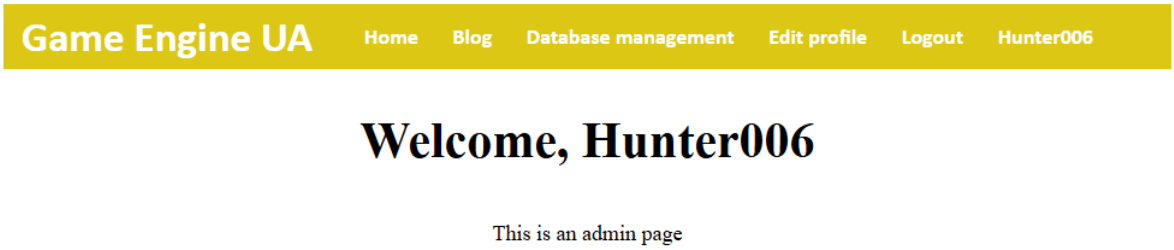
Б.7. Сторінка публікації



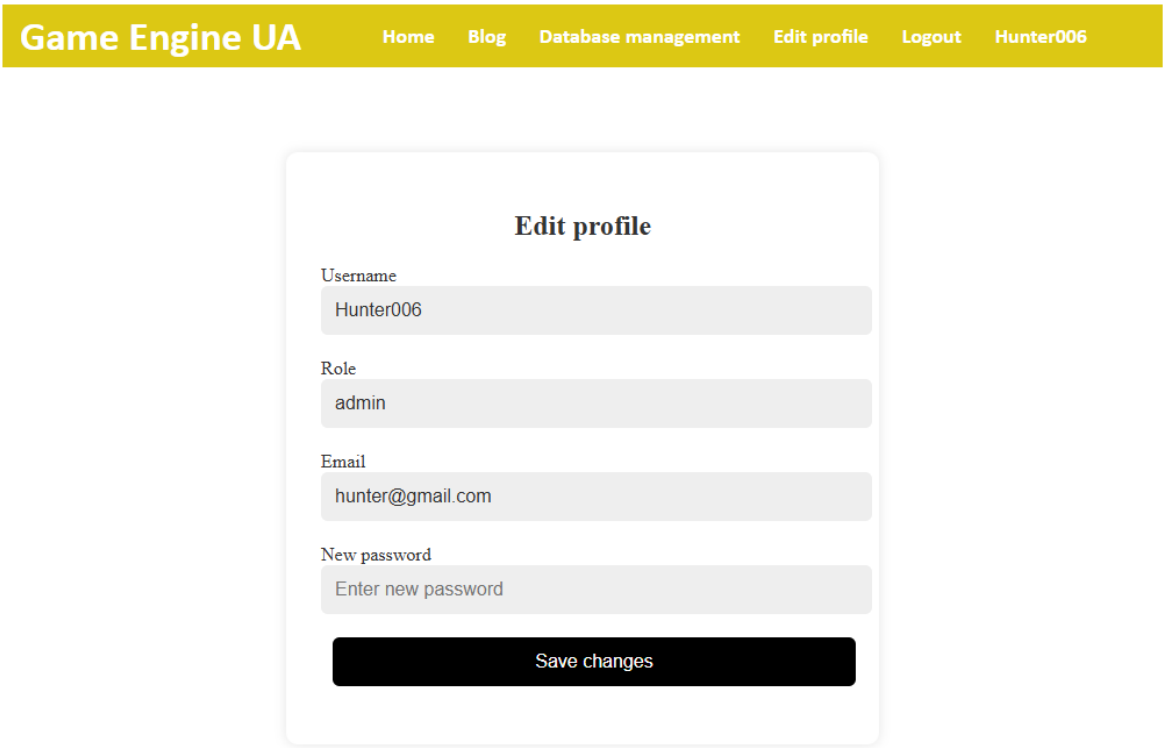
Б.8. Сторінка редагування публікації



Б.9. Головна сторінка адміністратора



Б.10. Сторінка редагування профілю адміністратора



Б.11. Сторінка панель керування базою даних