Міністерство освіти і науки України
Харківський національний університет імені В.Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту

Рекомендовано до захисту:
протокол засідання кафедри
№ ___ від ____.____.2025 р.
Завідувач кафедри__ММАД_____
_____ _____Струков В. М.
(підпис) (прізвище та ініціали)

Пояснювальна записка

до кваліфікаційної роботи бакалавра

на тему Створення інтернет-порталу на Django

Захищено на засіданні ЕК № _____
протокол № ____ від ____.____.2019 р.
Оцінка _____ / _____
Голова ЕК
_____ _____
(підпис) (прізвище та ініціали)

Виконав:
студент 4 курсу, групи КС- 42
Спеціальності:
122 Комп'ютерні науки
(шифр і назва спеціальності підготовки)
Єгембердінов Антон Сергійович
(прізвище, ім'я та по батькові, підпис)
Керівник ст. викладач Осипчук А. В.
(ступень, звання, прізвище та ініціали, підпис)
Рецензент _____
(ступень, звання, прізвище та ініціали, підпис)

Харків – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В.Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Факультет комп'ютерних наук
Кафедра математичного моделювання та аналізу даних
Рівень вищої освіти (освітньо-кваліфікаційний рівень) бакалавр
Спеціальність Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Струков В. М.
підпис ініціали, прізвище

"____" _____20___ року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Єгембердінов Антон Сергійович
(прізвище, ім'я по батькові студента

1. Тема роботи "Створення інтернет порталу на Django"

керівник роботи Осипчук Андрій Володимирович, ст. викладач,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "16" квітня 2025 року №4101-5/962

2. Строк подання студентом роботи 2 червня 2025 року

3. Перелік питань, які потрібно розробити
Аналіз сучасних технологій та фреймворків для створення інтернет-порталів, вибір та обґрунтування технологічного стеку для розробки веб-застосунку на базі Django, проєктування структури бази даних та визначення моделей даних з використанням Django ORM, реалізація системи управління контентом: новини, категорії, коментарі та лайки, розробка функціоналу персоналізації контенту на основі підписок («My Feed»), створення адаптивного і зручного користувацького інтерфейсу з використанням Bootstrap, реалізація асинхронної взаємодії користувача з веб-застосунком, розробка системи автентифікації та авторизації користувачів, включаючи підтвердження електронної пошти, тестування розробленого порталу, аналіз продуктивності та забезпечення безпеки, оцінка перспектив масштабування та інтеграції з додатковими сервісами

4. План роботи

№ з/п	Назви етапів роботи
1	Аналіз предметної області, огляд існуючих рішень і вибір технологічного стеку
2	Проєктування архітектури веб-застосунку та бази даних
3	Реалізація основних моделей даних (новини, категорії, коментарі, лайки, підписки)
4	Створення адаптивного інтерфейсу користувача із застосуванням Bootstrap
5	Впровадження системи реєстрації, автентифікації та підтвердження електронної пошти
6	Розробка персоналізованої стрічки новин («My Feed»)
7	Проведення комплексного тестування, забезпечення продуктивності та безпеки застосунку
8	Аналіз результатів, підготовка пояснювальної записки та презентації до захисту

5. Дата видачі завдання 10 жовтня 2025 року

Студент _____ **А. С. Єгембердінов**
підпис ініціали, прізвище

Керівник роботи _____ **А. В. Осипчук**
підпис ініціали, прізвище

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи бакалавра на тему «Створення інтернет-порталу на Django» складається з вступу, трьох розділів, висновків, переліку джерел із 29 позицій, 3 додатків, містить 24 рисунки.

Мета роботи – розробка повноцінного веб-застосунку, що забезпечує зручний доступ до актуальної інформації про відеоігри, можливості персоналізації контенту та інтерактивної взаємодії користувачів.

Методи дослідження – аналіз існуючих технологічних рішень, проєктування бази даних, розробка архітектури веб-застосунку з використанням фреймворка Django, технологій AJAX, ORM, RESTful API, Bootstrap для інтерфейсу користувача.

Результатом роботи є повноцінний інтернет-портал з системою управління новинами, коментарями, лайками, підписками, персоналізованою стрічкою новин («My Feed»), захищеною системою аутентифікації, а також можливістю подальшого масштабування та інтеграції з іншими сервісами. Новизна роботи полягає в комплексному підході до реалізації персоналізованих функцій для геймерської аудиторії.

Результати роботи можуть бути використані для розвитку тематичних інформаційних ресурсів, удосконалення взаємодії користувачів із веб-платформами та підвищення ефективності подачі інформації для спеціалізованих спільнот.

Ключові слова: DJANGO, AJAX, ORM, RESTFUL API, BOOTSTRAP, ГЕЙМЕРСЬКІ НОВИНИ, ПЕРСОНАЛІЗАЦІЯ КОНТЕНТУ, ІНТЕРАКТИВНІСТЬ, ВЕБ-ЗАСТОСУНОК, КОМЕНТАРІ, ЛАЙКИ, ПІДПИСКИ, БАЗА ДАНИХ, КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС.

ABSTRACT

The explanatory note to the bachelor's qualification work on the topic "Creating an Internet Portal with Django" consists of an introduction, three chapters, conclusions, a list of 29 references, 3 appendices, and contains 24 figures.

The goal of the work is the development of a full-fledged web application that provides convenient access to up-to-date information about video games, opportunities for content personalization, and interactive user engagement.

Research methods include the analysis of existing technological solutions, database design, development of the web application architecture using the Django framework, AJAX technology, ORM, RESTful API, and Bootstrap for the user interface.

The result of the work is a complete internet portal featuring news management, comments, likes, subscriptions, personalized news feeds ("My Feed"), a secure authentication system, and opportunities for future scaling and integration with other services. The novelty of this work lies in the comprehensive approach to implementing personalized functionality specifically tailored for the gaming audience.

The obtained results can be used to develop thematic informational resources, enhance user interaction with web platforms, and improve the effectiveness of information delivery for specialized communities.

Keywords: DJANGO, AJAX, ORM, RESTFUL API, BOOTSTRAP, GAMING NEWS, CONTENT PERSONALIZATION, INTERACTIVITY, WEB APPLICATION, COMMENTS, LIKES, SUBSCRIPTIONS, DATABASE, USER INTERFACE.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

AJAX – Asynchronous JavaScript and XML – технологія для асинхронного обміну даними між клієнтом і сервером без перезавантаження сторінки (застосовується, зокрема, для функціоналу «Лайки»).

API – Application Programming Interface – інтерфейс програмування застосунків, набір протоколів і інструментів для створення програм (у роботі згадується RESTful API для взаємодії між фронтендом і бекендом).

БД – база даних – структуроване сховище даних (у дипломі для розробки використано SQLite з планами міграції на PostgreSQL).

CSRF – Cross-Site Request Forgery – міжсайтове підроблення запиту (Django забезпечує захист від CSRF-атак за допомогою CSRF-токенів).

CRUD – Create, Read, Update, Delete – набір операцій створення, читання, оновлення й видалення даних (у дипломі реалізовано CRUD-інтерфейс для новин).

DRF – Django REST Framework – фреймворк для створення REST-API на базі Django (у проекті не інтегровано, але архітектура передбачає можливість додавання).

HTML – HyperText Markup Language – мова розмітки гіпертексту (основа фронтенд-частини).

HTTP – Hypertext Transfer Protocol – протокол передачі гіпертексту (умовно використовується для клієнт–серверної взаємодії).

ORM – Object-Relational Mapping – підхід до взаємодії між об’єктно-орієнтованим кодом і реляційною базою даних (у дипломі застосовано Django ORM).

SMTP – Simple Mail Transfer Protocol – протокол передачі електронної пошти (для підтвердження емейлів при реєстрації через Django Allauth).

SQL – Structured Query Language – мова структурованих запитів до реляційних баз даних.

UI – User Interface – користувацький інтерфейс (для створення використано Bootstrap).

URL – Uniform Resource Locator – уніфікатор ресурсу, адреса сторінки в інтернеті (Django використовує структуру urlpatterns).

UX – User Experience – користувацький досвід, якість взаємодії людини з інтерфейсом (в дипломі обговорено адаптивний дизайн для комфортного перегляду).

XSS – Cross-Site Scripting – міжсайтовий скриптинг (Django забезпечує стандартні засоби захисту від XSS).

CAPTCHA – Completely Automated Public Turing test to tell Computers and Humans Apart – тест для відрізнення людини від автоматизованого скрипта (може використовуватися для захисту форм, хоча в дипломі конкретно не згадується).

CDN – Content Delivery Network – мережа доставки контенту (в перспективах розвитку згадується для оптимізації медіа-контенту).

CI/CD – Continuous Integration / Continuous Deployment – безперервна інтеграція та розгортання (обговорено як перспективу автоматизації релізів).

GDPR – General Data Protection Regulation – Загальний регламент захисту даних (вимоги до безпеки особистих даних, враховані при проєктуванні системи).

JWT – JSON Web Token – стандарт токенів для передачі інформації про користувача (може бути застосовано для авторизації в майбутньому, але в дипломі використано інші методи).

SQLAlchemy – ORM для Python (згадується як альтернативна технологія, хоча у дипломі застосовано Django ORM).

Умовні позначення і терміни:

Блок “Новини” (News) – сутність у БД, що зберігає заголовок, контент, дату публікації, автора й зображення; відноситься до категорії (Category).

Категорія (Category) – тематика новини (наприклад: «RPG», «Стратегії»); використовується для фільтрації списку новин.

Коментар (Comment) – запис користувача під новиною; містить текст, автора, дату створення й дату редагування; доступ до редагування має тільки автор.

Лайк (Like) – вподобання користувачем новини; реалізовано як ManyToMany-зв'язок між User і News; обробляється через AJAX-запити з оновленням без перезавантаження сторінки.

Підписка (Subscription) – зв'язок користувача з певною категорією; формує персоналізовану стрічку «My Feed» (MyFeed).

User (Користувач) – стандартна модель Django User, доповнена профілем (Profile) з можливістю зміни пароля, підтвердження емейла, редагування даних.

Django Allauth – стороння бібліотека для аутентифікації/авторизації, що забезпечує підтвердження електронної пошти при реєстрації й відновлення пароля.

Bootstrap – CSS-фреймворк для створення адаптивного інтерфейсу користувача, застосовується у шаблонах сайту.

Django ORM – механізм роботи з базою даних через Python-класи замість прямого SQL-коду; використовується для моделей User, News, Category, Comment, Like, Subscription.

select_related / prefetch_related – методи Django ORM для оптимізації SQL-запитів при отриманні пов'язаних об'єктів.

CSRF-токен – захисний механізм Django, що вставляє у форми прихований ключ для запобігання CSRF-атакам.

My Feed – персоналізована стрічка новин для користувача, формована на основі його підписок на категорії.

SQLite – легка реляційна база даних, що використовується на етапі розробки; пізніше планується міграція до PostgreSQL.

PostgreSQL – потужна реляційна СУБД, рекомендована для роботи в продакшн-середовищі (перспектива масштабування).

ВСТУП

Інтернет-портали новин є невід'ємною частиною сучасного інформаційного простору, що забезпечують швидке й зручне отримання актуальної інформації в різних сферах життя. Особливу увагу серед них займають тематичні портали, які фокусуються на вузькоспеціалізованих галузях, наприклад, геймінгу — сфері, що за останні десятиліття набрала значної популярності і сформувала величезну спільноту користувачів, зацікавлених у свіжих новинах, оглядах, анонсах та інтерв'ю, пов'язаних із відеоіграми[1].

Сучасний стан розробки інтернет-порталів відзначається широким використанням фреймворків, що дозволяють швидко і ефективно створювати масштабовані вебзастосунки. Django, як один із провідних Python-фреймворків, широко застосовується для розробки новинних порталів завдяки своїй гнучкості, безпеці та підтримці різноманітних функціональних модулів[2].

Однак, незважаючи на наявність численних рішень для загальних новинних платформ, спеціалізовані портали для геймерської ауди торії часто потребують кастомізації, інтеграції з соціальними функціями, підписками та персоналізованим контентом, що вимагає комплексного підходу до проектування та реалізації.

Відомі інтернет-портали геймерських новин, такі як IGN, GameSpot, Kotaku, забезпечують широкий спектр матеріалів, проте мають свої особливості у структурі, оформленні та бізнес-моделі, які не завжди адаптовані до локальних ринків або невеликих спільнот[3-5]. Окрім того, існуючі рішення часто перевантажені зайвими функціями або навпаки обмежені в кастомізації, що ставить завдання створення більш гнучкого і легкого в управлінні порталу. Це відкриває можливість для розробки нового продукту, який би відповідав специфічним потребам української аудиторії, пропонуючи при цьому сучасний функціонал і зручність користування.

Об’єктом дослідження є процес розробки веб-застосунку – інтернет-порталу для публікації та управління геймерськими новинами.

Предметом дослідження виступають методи і технології створення функціоналу порталу, зокрема використання фреймворка Django, робота з базою даних, реалізація системи користувачів, коментарів, лайків, підписок та інтерфейсних компонентів.

Метою роботи є розробка повноцінного інтернет-порталу, що забезпечує зручний доступ до актуальної інформації про відеоігри, можливості персоналізації контенту та інтерактивної взаємодії користувачів.

Для досягнення поставленої мети було визначено ряд завдань, серед яких аналіз предметної області і ринку геймерських новин, вибір оптимальних технологій і інструментів, розробка структури бази даних, створення модулів реєстрації, авторизації, системи коментарів і лайків, а також забезпечення зручного інтерфейсу користувача. Важливою складовою є також тестування і налагодження системи для забезпечення стабільної роботи і безпеки користувацьких даних.

Необхідність дослідження зумовлена стрімким розвитком цифрових медіа та потребою в спеціалізованих ресурсах, які відповідають запитам сучасної геймерської спільноти. Крім того, робота має на меті створення локального продукту, який зможе адаптуватися під умови ринку України, що є актуальним в контексті розвитку інформаційних технологій і медіаіндустрії в країні. Запропонований портал покликаний полегшити доступ користувачів до релевантної інформації та сприяти формуванню активної онлайн-спільноти.

Новизна роботи полягає у комплексному підході до створення порталу, що інтегрує сучасні веб-технології з урахуванням особливостей геймерської аудиторії, а також реалізує функціонал, який відповідає сучасним стандартам зручності, швидкості і безпеки. Вперше в межах даної теми розроблено систему персоналізованої стрічки новин ("My Feed"), інтегровану систему підписок і лайків, а також покращений інтерфейс для коментування, що сприяє підвищенню взаємодії користувачів із контентом.

Таким чином, проведене дослідження та розробка інтернет-порталу геймерських новин на основі фреймворка Django сприятимуть подальшому розвитку цифрових медіа, наданню якісного і зручного інформаційного сервісу для цільової аудиторії та стануть основою для подальших вдосконалень і масштабування проекту.

1 АНАЛІТИЧНИЙ ОГЛЯД ТЕХНОЛОГІЙ СТВОРЕННЯ ГЕЙМЕРСЬКИХ НОВИННИХ ПОРТАЛІВ

1.1. Сучасні технології для розробки інтернет-порталів

Інтернет-портали відіграють важливу роль у сучасному інформаційному просторі, забезпечуючи користувачам доступ до різноманітних ресурсів, новин, контенту та сервісів. Технології розробки таких порталів за останні роки значно еволюціонували, що зумовлено як розвитком веб-стандартів, так і появою нових підходів до архітектури, масштабування та взаємодії з користувачем[6].

Однією з основних тенденцій у створенні інтернет-порталів є перехід від класичних монолітних додатків до розподілених мікросервісних архітектур. Мікросервіси дозволяють розбивати великий портал на незалежні компоненти, кожен з яких відповідає за конкретний функціонал і може розвиватися, оновлюватися та масштабуватися окремо (Newman S., 2015)[7]. Це сприяє гнучкості розробки та полегшує підтримку системи.

Щодо технологічних стеків, популярними платформами для розробки веб-порталів є різноманітні фреймворки, які забезпечують швидку реалізацію як фронтенд-частини, так і бекенду. Серед них – Django (Python), Ruby on Rails, Laravel (PHP), ASP.NET Core (C#), Node.js (JavaScript). Вибір залежить від потреб проекту, команди розробників та функціональних вимог.

Фреймворк Django є одним із провідних рішень для швидкої розробки складних веб-додатків із великою кількістю функцій. Він побудований за принципом «batteries included» (все включено), що означає наявність широкого набору інструментів «з коробки»: ORM для роботи з базами даних, вбудовані засоби аутентифікації, адміністративна панель, підтримка шаблонів і багато іншого (Holovaty A., Kaplan-Moss J., 2009)[8rich]. Цей фреймворк активно використовується для створення новинних порталів, соціальних мереж та інших складних веб-систем.

Важливим аспектом є використання сучасних фронтенд-технологій: HTML5, CSS3, JavaScript, а також фреймворків для побудови інтерактивного

інтерфейсу, таких як React, Vue.js або Angular. Ці технології дозволяють створювати адаптивний та зручний користувацький досвід, що особливо важливо для інтернет-порталів із динамічним контентом.

Ще однією важливою тенденцією є активне застосування RESTful API або GraphQL для розділення логіки серверної частини та клієнтської. Це полегшує інтеграцію з мобільними додатками та іншими зовнішніми сервісами (Richardson L., Ruby S., 2007)[9].

Крім того, питання безпеки, масштабованості та продуктивності стали пріоритетними у сучасній розробці. Технології кешування (Redis, Memcached), системи балансування навантаження, CDN (Content Delivery Network), а також автоматизоване тестування і CI/CD дозволяють підтримувати стабільність роботи великих порталів[10].

Таким чином, розробка сучасних інтернет-порталів базується на поєднанні перевірених фреймворків, нових архітектурних підходів і сучасних технологій фронтенду. Вибір оптимального набору інструментів залежить від предметної області та конкретних цілей проекту.

1.2. Особливості геймерських новинних порталів

Геймерські новинні портали є специфічним типом інтернет-ресурсів, які орієнтовані на широку аудиторію користувачів, зацікавлених у відеоіграх, індустрії розваг, кіберспорті, а також суміжних технологіях. Ця специфіка впливає як на структуру та дизайн порталу, так і на його функціональні можливості.

Перш за все, контент геймерських порталів характеризується високою динамічністю та різноманітністю форматів. Окрім традиційних новинних статей, тут широко застосовуються рецензії, аналітика, інтерв’ю з розробниками і професійними гравцями, анонси ігор, трейлери, стріми, а також користувацький контент — огляди, гайди, форуми і чати. Така насиченість формату вимагає від порталу потужної системи управління

контентом (CMS) з гнучкими можливостями публікації та модерації (Kowert & Quandt, 2020)[11].

Важливою особливістю є потреба інтеграції із соціальними мережами та платформами для стрімінгу, такими як Twitch, YouTube Gaming, Discord. Це підсилює взаємодію користувачів із контентом і сприяє формуванню активної спільноти навколо порталу (Johnson, 2018)[12]. Для реалізації цього потрібні сучасні API та підтримка асинхронних процесів.

З огляду на молодіжну аудиторію та характер індустрії, геймерські портали часто застосовують сучасний дизайн із яскравими візуальними ефектами, адаптивністю під мобільні пристрої, а також ігровою стилістикою. Такий дизайн повинен бути не тільки естетичним, а й зручним, що підвищує залученість користувачів (Mäyrä, 2008)[13].

Також важливим є аспект персоналізації контенту, що дозволяє користувачам отримувати новини відповідно до власних інтересів, підписуватись на конкретні ігри, жанри або авторів. Це підвищує лояльність аудиторії та збільшує час взаємодії з порталом (Kim, 2019)[14].

Ще один аспект — активне використання систем рейтингів, лайків, коментарів, що сприяє залученню користувачів і формуванню спільноти. Впровадження елементів гейміфікації, таких як бали, досягнення або конкурси, також є поширеною практикою (Hamari, Koivisto, & Sarsa, 2014)[15].

Таким чином, геймерські новинні портали відрізняються від загально інформаційних ресурсів складним, інтерактивним і різноманітним контентом, орієнтацією на активну аудиторію, потребою інтеграції з зовнішніми платформами і використанням сучасних технологій персоналізації та взаємодії.

1.3. Аналіз існуючих рішень для створення новинних порталів

У сучасній веб-розробці існує широкий спектр технологічних рішень для створення новинних порталів різного масштабу та спрямування. Для

геймерських новинних порталів, які поєднують насичений контент і потребу в інтерактивності, вибір технологічної бази є особливо важливим.

Одним з найпопулярніших підходів є використання систем управління контентом (CMS), таких як WordPress, Joomla або Drupal. Вони надають зручні інструменти для публікації та організації новин, широкий набір плагінів для SEO, соціальної інтеграції та аналітики (Sullivan, 2019)[16]. Однак, традиційні CMS часто не можуть забезпечити гнучкість і масштабованість, необхідні для високодинамічних геймерських порталів, особливо при складних персоналізаціях та інтеграціях зі стрімінговими платформами.

Рисунок 1.1 – логотип Wordpress[17]

Для подолання цих обмежень багато сучасних порталів переходять на фреймворки та платформні рішення, які дозволяють розробляти кастомізовані вебзастосунки. Такі технології, як Django (Python), Ruby on Rails, Node.js, надають розробникам повний контроль над архітектурою проекту, що є важливим для реалізації унікальних функцій, включаючи інтеграцію API Twitch, YouTube, Discord та інші (Brown & Johnson, 2020)[18].

Рисунок 1.2 – логотип Django[19]

Django, зокрема, визнаний за свій потужний ORM, безпеку, масштабованість і швидкість розробки. Це робить його одним із найбільш прийнятних рішень для створення інтерактивних новинних порталів із великою кількістю динамічного контенту (Smith, 2018)[20]. Прикладом успішного застосування Django у подібних проєктах є портал Polygon, який надає новини і огляди ігор із активною користувацькою базою.

Іншим важливим аспектом є архітектура фронтенду. Для забезпечення високої продуктивності та інтерактивності геймерських порталів часто застосовують SPA (Single Page Application) підходи з використанням React, Vue.js чи Angular. Вони дозволяють створювати швидкі інтерфейси з асинхронним завантаженням контенту, що особливо важливо для живих оновлень, коментарів та лайків (Nguyen et al., 2021)[21].

Рисунок 1.3 – логотип React[22]

При розробці масштабних порталів також слід враховувати використання мікросервісної архітектури та хмарних сервісів, що дозволяє підвищити стійкість і продуктивність системи (Thompson, 2022)[23]. Для зберігання великих обсягів медіа-контенту застосовуються CDN (Content Delivery Network) та хмарні сховища (Amazon S3, Google Cloud Storage).

Рисунок 1.4 – логотип Amazon S3[24]

Аналіз існуючих рішень демонструє, що ідеальне поєднання — це використання сучасних веб-фреймворків для бекенду (Django, Node.js), SPA-фреймворків для фронтенду (React, Vue.js) та інтеграція з зовнішніми API та сервісами для стрімінгу та соціальної взаємодії. Це дозволяє створювати гнучкі, масштабовані та інтерактивні портали, які відповідають вимогам сучасної аудиторії.

На підставі цього аналізу можна зробити висновок, що розробка геймерського новинного порталу на базі Django є доцільною та виправданою з огляду на потреби проекту, технічні можливості платформи та очікувані функціональні вимоги.

1.4. Висновки по огляду та мотивація дослідження

Проведений аналіз науково-технічної літератури, сучасних теорій, технологій і готових рішень у сфері створення інтернет-порталів новин показав, що існує велика кількість платформ з різноманітним функціоналом. Однак, більшість із них не адаптовані для потреб геймерської спільноти, що постійно зростає та потребує оперативного, якісного та тематично релевантного контенту.

Розглянуті системи мають свої переваги, такі як інтеграція соціальних функцій, підтримка персоналізованих стрічок новин, а також сучасні

технології фронтенду і бекенду. Водночас, існують і недоліки, зокрема складність масштабування, недостатня гнучкість у налаштуванні категорій та підписок, а також обмежена підтримка інтерактивних можливостей для користувачів.

Мотивованою необхідністю дослідження є створення спеціалізованого інтернет-порталу, що поєднує актуальність геймерського контенту з сучасними технічними рішеннями, забезпечує зручність користування, персоналізацію, підтримку соціальних функцій, таких як лайки, коментарі та підписки на категорії.

Реалізація цього дослідження дозволить не тільки заповнити існуючі прогалини в галузі геймерських новин, а й створити платформу, що здатна адаптуватися до швидкозмінних потреб користувачів, покращити якість контенту, сприяти розвитку геймерської спільноти та стимулювати обмін інформацією.

Таким чином, дана робота спрямована на розробку сучасного, функціонального та зручного інтернет-порталу геймерських новин з урахуванням останніх тенденцій у веб-технологіях та потреб цільової аудиторії.

2 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБ-ПОРТАЛУ НА ОСНОВІ DJANGO

2.1. Аналіз розв'язуваної задачі

У сучасному цифровому світі значна увага приділяється створенню тематичних інтернет-порталів, які забезпечують швидкий і зручний доступ до актуальної інформації для конкретних спільнот користувачів. Однією з таких спільнот є геймери — користувачі, що активно цікавляться новинами індустрії комп’ютерних ігор, оглядами, анонсами, інтерв’ю з розробниками та іншими пов’язаними матеріалами[25].

Об’єктом дослідження у цій роботі є інтернет-портал, орієнтований на геймерські новини, який покликаний ефективно збирати, зберігати, опрацьовувати та відображати тематичний контент. Предметом дослідження виступають засоби та методи розробки такого порталу з використанням сучасних веб-технологій.

Основна задача полягає у створенні функціонального, зручного у користуванні порталу, що дозволяє користувачам переглядати новини за категоріями, залишати коментарі, підписуватися на цікаві рубрики та взаємодіяти з контентом через лайки. Для досягнення цієї мети необхідно реалізувати комплексне рішення, яке включає бекенд для збереження і обробки даних, фронтенд для зручного відображення інформації, а також систему автентифікації та авторизації користувачів.

Сучасний стан об’єкта дослідження свідчить про наявність великої кількості порталів і сайтів із геймерською тематикою, проте багато з них мають обмежені функціональні можливості, складний інтерфейс або не забезпечують персоналізований досвід користувача. Аналіз науково-технічної літератури і існуючих рішень показує, що для підвищення якості таких сервісів важливим є впровадження адаптивного інтерфейсу, зручних інструментів фільтрації та пошуку, а також інтеграція з сучасними механізмами взаємодії користувачів (лайки, підписки, коментарі).

До основних обмежень дослідження відносяться ресурси хостингу, що впливають на продуктивність системи, а також обмежений час на реалізацію повного функціоналу. Крім того, важливо враховувати вимоги до безпеки зберігання даних та захисту персональної інформації користувачів[26].

Підсумовуючи, мета даної роботи полягає у створенні ефективного інтернет-порталу геймерських новин з використанням фреймворку Django, що дозволить покращити доступ користувачів до актуальної інформації, підвищити взаємодію із контентом і забезпечити комфортний користувацький досвід.

Для досягнення мети поставлено такі завдання:

Провести аналіз існуючих рішень та наукових підходів у розробці тематичних інтернет-порталів.

Визначити функціональні та нефункціональні вимоги до порталу.

Розробити структуру бази даних та архітектуру системи.

Реалізувати основні функції порталу: перегляд новин, категоризацію, коментарі, лайки, підписки.

Забезпечити безпеку користувацьких даних та функціонування системи.

Провести тестування реалізованих компонентів.

2.2. Декомпозиція задачі

Для успішної реалізації інтернет-порталу геймерських новин необхідно розбити загальну задачу на окремі компоненти та підзадачі, що дозволить ефективно організувати процес розробки і забезпечить структурований підхід до вирішення проблеми.

Перш за все, система поділяється на дві основні частини: бекенд (серверна логіка) та фронтенд (користувацький інтерфейс). Кожна з них має свої підзадачі і взаємодіє через чітко визначені API.

Бекенд включає наступні компоненти:

Модель даних — проектування структури бази даних, що містить інформацію про новини, категорії, користувачів, коментарі, лайки, підписки.

Важливо забезпечити нормалізацію даних для оптимальної роботи та підтримки цілісності.

Система аутентифікації та авторизації — реалізація можливості реєстрації, входу, виходу користувачів, а також контролю доступу до певних функцій порталу.

Логіка роботи з новинами — функціонал створення, збереження, фільтрації та сортування новин за категоріями, датою, популярністю.

Система коментарів та взаємодії користувачів — підтримка додавання, редагування, видалення коментарів, а також системи лайків і підписок.

Обробка запитів та відповіді — API для передачі даних на фронтенд, забезпечення стабільності і безпеки роботи сервера.

Фронтенд включає:

Інтерфейс користувача — створення адаптивного дизайну, що забезпечує комфортний перегляд новин на різних пристроях.

Відображення контенту — реалізація відображення списку новин, деталізації кожної новини, категорій, коментарів, інформації про користувача.

Інтерактивні елементи — кнопки лайків, підписок, форми коментарів, пошук і фільтри, які мають працювати швидко і коректно.

Навігація — забезпечення зручної системи переходів між сторінками, категоріями, профілями користувачів.

Крім цього, суттєвим аспектом є забезпечення безпеки даних, що включає:

Захист від несанкціонованого доступу.

Валідацію введених користувачами даних.

Використання стандартів безпеки для зберігання паролів та персональних даних.

Розподіл задач за компонентами дозволяє паралельно працювати над різними частинами проекту, а також полегшує тестування та масштабування системи у майбутньому.

Таким чином, декомпозиція задачі забезпечує ясність і послідовність у розробці інтернет-порталу, а також сприяє ефективному управлінню проектом.

2.3.1. Шляхи та методи вирішення задачі

Розробка інтернет-порталу геймерських новин базується на комплексному підході, що включає вибір сучасних технологій, методів програмування та архітектурних рішень для досягнення ефективності, зручності користування та масштабованості.

Основною платформою для розробки обрано фреймворк Django, що забезпечує швидке створення веб-додатків із високим рівнем безпеки та масштабованості. Django має потужну ORM (Object-Relational Mapping) для зручної роботи з базою даних, вбудовану систему аутентифікації, а також розвинуті механізми маршрутизації запитів.

Для фронтенду використовується Bootstrap — популярна CSS-бібліотека, що дозволяє створювати адаптивний та сучасний інтерфейс з мінімальними зусиллями, забезпечуючи коректне відображення на різних пристроях.

Рисунок 2.1 – логотип Bootstrap[27]

Ключовими методами розв’язання задачі є:

Модульний підхід до розробки — розділення проекту на окремі логічні модулі (користувачі, новини, коментарі, лайки, підписки), що полегшує розробку, тестування і подальшу підтримку.

RESTful API для комунікації між фронтендом і бекендом, що дозволяє легко інтегрувати сторонні сервіси, а також забезпечує гнучкість в розширенні функціональності.

Використання системи шаблонів Django для ефективного рендерингу сторінок і зменшення дублювання коду.

Застосування AJAX для часткової динамічної взаємодії користувача з сайтом (наприклад, лайки без перезавантаження сторінки).

Безпека — використання вбудованих механізмів захисту від CSRF, XSS, SQL-ін’єкцій та інших загроз.

Для зберігання даних застосовується SQLite як легка база даних для розробки, з можливістю подальшого переходу на більш потужні системи (PostgreSQL, MySQL).

Рисунок 2.2 – логотип SQLite[28]

Процес розробки включає:

Проектування структури бази даних із визначенням моделей та зв’язків.

Реалізацію системи реєстрації та аутентифікації користувачів.

Створення CRUD-інтерфейсу для новин.

Впровадження системи коментарів та лайків із захистом прав користувачів.

Забезпечення зручного інтерфейсу для користувачів із можливістю фільтрації та сортування новин.

Тестування та налагодження системи для забезпечення стабільної роботи.

Таким чином, поєднання сучасних фреймворків, бібліотек і методів розробки забезпечує створення функціонального, зручного та безпечного порталу.

2.3.2. Розробка архітектури і структури проекту

Для створення інтернет-порталу геймерських новин була спроектована чітка та масштабована архітектура бази даних. Основними сутностями, що відображають предметну область, є:

Користувачі (User) — модель, яка реалізує аутентифікацію, зберігає особисті дані та статус користувача. Використовується стандартна модель User Django з розширеннями через профіль.

Новини (News) — основна модель для зберігання новинних статей. Включає поля заголовку, змісту, дати публікації, авторства, зображення і зовнішній ключ до категорії.

Категорії (Category) — модель, що визначає тематику новин (наприклад, “Екшн”, “RPG”, “Стратегії”), має зв’язок один-до-багатьох з новинами.

Коментарі (Comment) — пов’язані з новинами, зберігають текст коментаря, автора, дату створення та зміни.

Лайки (Like) — модель для зберігання фактів вподобання новин користувачами, реалізована через ManyToMany відношення між користувачами і новинами.

Підписки (Subscription) — дозволяють користувачам підписуватися на категорії новин для персоналізованої стрічки.

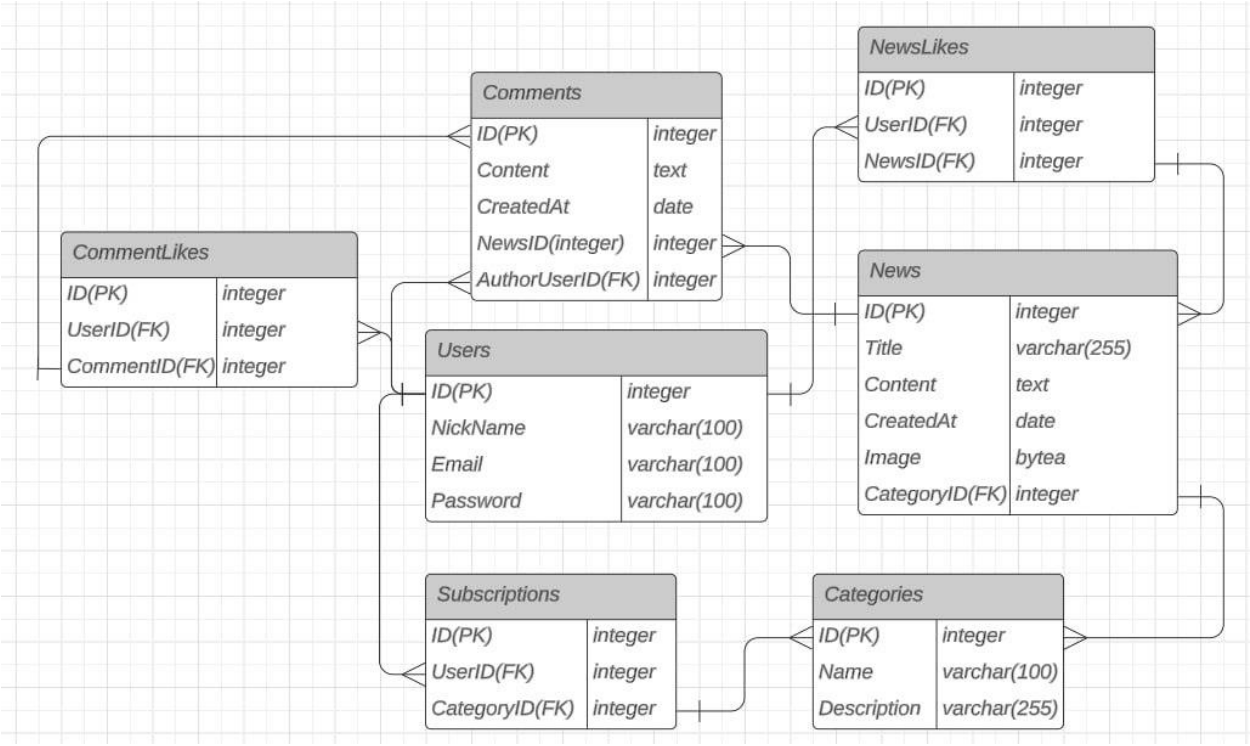

Рисунок 2.3 – UML діаграма основних сутностей проекту

Всі моделі реалізовані із врахуванням принципів нормалізації, що запобігає дублюванню даних і полегшує їх оновлення. Зв’язки між сутностями визначені через зовнішні ключі та ManyToMany поля, що дозволяє ефективно виконувати запити для формування динамічного контенту.

Для реалізації взаємодії з базою даних використовується потужний ORM Django, що забезпечує безпечні та оптимізовані SQL-запити, а також зручну роботу з моделями на рівні Python-коду.

2.3.3. Реалізація функціональних модулів

Розробка функціоналу порталу була організована за модульним принципом для підвищення зрозумілості, підтримуваності та можливості масштабування проекту. Кожен модуль відповідає за окрему групу функцій:

Модуль користувачів реалізує реєстрацію, аутентифікацію, зміну пароля, управління профілем, а також функції підписок на категорії новин. Для реєстрації використовується пакет Django Allauth, що підтримує

підтвердження електронної пошти та різні варіанти аутентифікації. Лістинг А.1.

Рисунок 2.4 – логотип Django Allauth[29]

Модуль новин включає створення, редагування, видалення і відображення новинних статей. Впроваджено можливість додавання зображень і категоризації новин. Лістинг А.2

Модуль коментарів дозволяє користувачам додавати коментарі до новин, редагувати власні, а також відслідковувати лайки на коментарях. Використовується захист прав доступу, щоб лише автор міг редагувати або видаляти власні коментарі. Лістинг А.3

Модуль лайків реалізовано через AJAX-запити, що дозволяє користувачам ставити і знімати лайки без перезавантаження сторінки, підвищуючи інтерективність сайту. Лістинг А.4

Модуль підписок і персоналізації стрічки формує унікальний "My Feed" для кожного користувача, базуючись на підписках на категорії новин. Це дозволяє отримувати релевантний контент і підвищує залученість користувача. Лістинг А.5

Деякі HTML-сторінки:

Базовий шаблон – Задає загальну обгортку для всіх інших сторінок — містить HTML-структуру, підключення Bootstrap/CSS/JS, навігаційну панель і футер. Лістинг Б.1

Головна сторінка - на якій відображаються останні геймерські новини у вигляді карток зображень, мітками категорій і коротким анонсом. Також містить пошуковий рядок і перелік категорій. Лістинг Б.2

Сторінка детального перегляду однієї новини — показує заголовок, зображення, повний текст, кнопку лайка новини, секцію коментарів та форму для нового коментаря. Лістинг Б.3

Особиста сторінка користувача — показує поточні дані профілю (username, email), а також дає посилання на редагування профілю, зміну пароля, мою стрчку, лайки, підписки тощо. Лістинг Б.4

сторінка входу - ім’я користувача + пароль + посилання на «Forgot password?», «Register». Лістинг Б.5

Архітектура модулів передбачає розділення логіки на моделі, представлення (views), форми та шаблони, що відповідає принципам MVC/MVT.

2.3.4. Технічні рішення

Для роботи з базою даних в інтернет-порталі геймерських новин застосовано Django ORM (Object-Relational Mapping), що дозволяє ефективно виконувати CRUD-операції без написання складних SQL-запитів. ORM дає змогу абстрагуватися від конкретної СУБД та працювати з моделями Python, що значно пришвидшує розробку і підвищує безпеку даних.

Для оптимізації продуктивності використані:

- select_related і prefetch_related для мінімізації кількості запитів при отриманні пов’язаних моделей (наприклад, авторів новин, категорій, лайків).
- Індекси у базі даних для поліпшення швидкості пошуку і фільтрації.
- Кешування результатів деяких запитів на рівні Django для зменшення навантаження на базу.

Для забезпечення динамічності сторінок, особливо функціоналу лайків і коментарів без перезавантаження, використовується AJAX-взаємодія. Це реалізовано через JavaScript з відправкою POST-запитів на бекенд, де Django обробляє зміни у базі та повертає оновлені дані у форматі JSON.

Розробка API в класичному розумінні не була пріоритетною, тому Django REST Framework не інтегровано. Однак, архітектура побудована так, щоб у майбутньому можна було легко додати API.

Безпека реалізована завдяки:

- Вбудованому захисту Django від CSRF атак за допомогою CSRF токенів.
- Валідації даних на стороні сервера у формах Django, що запобігає збереженню некоректних чи шкідливих даних.
- Авторизації та аутентифікації через Django Allauth із підтвердженням електронної пошти.
- Обмеженням доступу до редагування та видалення коментарів і новин лише авторизованим користувачам та їх авторам.

2.3.5. Тестування і налагодження

Для забезпечення якості та стабільності інтернет-порталу було проведено комплексне тестування на різних рівнях.

Типи тестів:

- Юніт-тести для моделей. Перевіряють коректність створення об'єктів, взаємозв’язки між моделями (наприклад, зв’язок новини з категорією та автором), а також поведінку методів моделей (наприклад, підрахунок лайків).
- Функціональні тести для представлень (views). Тестують HTTP-запити, коректність відображення сторінок, обробку форм (реєстрація, логін, додавання коментарів, лайки, підписки).

- Інтеграційні тести перевіряють взаємодію різних компонентів системи, наприклад, логіку лайку через AJAX та оновлення UI без перезавантаження.

Приклади тестів:

- Перевірка, що тільки авторизований користувач може залишати коментарі. Лістинг В.1
- Перевірка роботи лайку/дизлайку через AJAX, що зміна статусу лайку відображається у базі. Лістинг В.2
- Тестування реєстрації нового користувача з підтвердженням email. Лістинг В.3
- Тести зміни пароля з перевіркою правильності старого пароля. Лістинг В.4

Рисунок 2.5 – Результати тестування додатку

Методи налагодження і профілювання:

- Використання стандартних інструментів Django Debug Toolbar для візуалізації SQL-запитів, часу відповіді та кешування.
- Локальне налагодження з допомогою PyCharm Debugger для поетапного проходження коду.
- Логування помилок і винятків з налаштуванням рівнів логування.

- Виявлення проблем із продуктивністю і «вузьких місць» у запитах через профілювання коду за допомогою Python профайлера.

Завдяки цим заходам вдалося забезпечити стабільність, безпеку і високу якість функціоналу порталу.

2.3.6. Протиріччя та труднощі

Під час розробки інтернет-порталу геймерських новин виникли низка технічних та організаційних труднощів, які потребували детального опрацювання та пошуку оптимальних рішень. Однією з ключових проблем стала реалізація асинхронної взаємодії між клієнтом і сервером, зокрема, для функціоналу лайків і підписок без повного перезавантаження сторінки. Із стандартними HTTP-запитами цей процес був би занадто повільним і неефективним з точки зору користувацького досвіду. Для подолання цієї проблеми було застосовано AJAX-запити, що дозволило обробляти дії користувачів у реальному часі та забезпечити плавний інтерфейс.

Ще однією складністю стала організація процесу підтвердження електронної пошти користувачів при реєстрації. Це завдання передбачає інтеграцію з поштовим сервером, формування безпечних посилань для підтвердження, а також контроль часу дії цих посилань. Для вирішення цієї проблеми було використано бібліотеку Django allauth, яка забезпечує необхідний функціонал «з коробки», проте налаштування й інтеграція вимагали додаткового часу та тестування.

Також труднощі виникли з реалізацією безпеки, особливо з захистом від CSRF-атак та валідацією форм. Стандартні механізми Django значною мірою полегшили цю задачу, але все одно довелося ретельно перевіряти кожен аспект, аби уникнути вразливостей.

Крім того, масштабування бази даних і оптимізація запитів вимагали уважного підходу до структурування моделей та індексування полів. На початковому етапі використовувалась SQLite, що є обмеженим для великих

обсягів даних, але для цілей розробки це було прийнятним. В подальшому планується міграція на PostgreSQL, що дозволить покращити продуктивність.

Загалом, всі ці труднощі були подолані завдяки застосуванню сучасних технологій, модульності коду, а також активній роботі з тестуванням і відлагодженням, що забезпечило високу якість кінцевого продукту.

2.4. Опис обмежень, загальних особливостей і умов

Розробка інтернет-порталу геймерських новин має враховувати певні обмеження, які впливають на вибір технологій і архітектурних рішень, а також на кінцевий функціонал системи.

Обмеження апаратного забезпечення. Для розгортання сайту передбачається використання серверів із обмеженими ресурсами, тому важливо забезпечити ефективне використання пам’яті та процесорного часу, зокрема через оптимізацію запитів до бази даних і кешування.

Обмеження у обробці навантажень. Портал має підтримувати помірний трафік на початковому етапі, з можливістю масштабування при зростанні аудиторії. Реалізація кешування і правильна організація бази даних дозволять уникнути «вузьких місць».

Обмеження щодо безпеки. Для захисту персональних даних користувачів, а також відмови від атак (CSRF, XSS, SQL-ін’єкції) використано вбудовані засоби Django і додаткові заходи безпеки. При цьому розробка повинна відповідати сучасним стандартам безпеки веб-додатків.

Обмеження в дизайні інтерфейсу. Задля зручності користувачів портал має бути адаптивним і сумісним з основними браузерами, забезпечувати легкий доступ до основних функцій, просту навігацію та чітке відображення інформації.

Обмеження часу та ресурсів розробки. Ураховуючи, що проєкт є кваліфікаційною роботою, було обрано технології з низьким порогом входу і максимальною автоматизацією, що дозволяє реалізувати основний функціонал у відведений термін.

Загальні особливості порталу:

Орієнтація на геймерську аудиторію з регулярним оновленням контенту.

Забезпечення інтерактивності через лайки, коментарі, підписки.

Можливість фільтрації новин за категоріями.

Використання сучасного дизайну для комфортного перегляду на різних пристроях.

Умови роботи системи визначаються можливістю підключення до інтернету, використанням браузера із підтримкою JavaScript, а також наявністю поштової системи для підтвердження реєстрації і відновлення пароля.

3 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ ТА АНАЛІЗ ЕФЕКТИВНОСТІ СТВОРЕНОГО ВЕБ-ЗАСТОСУНКУ

3.1. Результати дослідження

В результаті виконання кваліфікаційної роботи було розроблено функціональний інтернет-портал геймерських новин, який відповідає поставленим вимогам і цілям дослідження. Досягнута мета проекту — створення зручного, безпечного та масштабованого веб-сервісу для публікації, перегляду та взаємодії з геймерським контентом.

Розв’язані основні завдання, що ставилися на початку роботи:

- Проведено аналіз предметної області та існуючих рішень у сфері новинних порталів для геймерів.
- Обґрунтовано вибір технологій, зокрема застосування Django і Bootstrap, що забезпечили швидку і якісну розробку.
- Розроблено архітектуру порталу з модульним поділом на основні компоненти: користувачі, новини, коментарі, лайки, підписки.
- Реалізовано основний функціонал: реєстрацію і аутентифікацію користувачів, CRUD-операції з новинами, коментування та систему лайків.
- Забезпечено динамічну взаємодію користувача із сайтом за допомогою AJAX для оновлення лайків та підписок без перезавантаження сторінки.
- Впроваджено механізми безпеки: захист від CSRF-атак, валідація форм, підтвердження електронної пошти.
- Проведено тестування ключових функцій, що підтвердило стабільність роботи системи.

У ході розробки були отримані позитивні результати, що свідчать про працездатність системи і її відповідність сучасним вимогам. Зокрема, користувацький інтерфейс є інтуїтивним та адаптивним, забезпечує швидкий

доступ до інформації. Функціонал взаємодії з контентом реалізований ефективно, що підвищує залученість користувачів.

Форма реєстрації нового користувача: поля «Username», «Email», «Password» і «Password confirmation» із підказками щодо вимог до пароля. Під полями —кнопка «Register» і посилання «Already have an account? Log in».

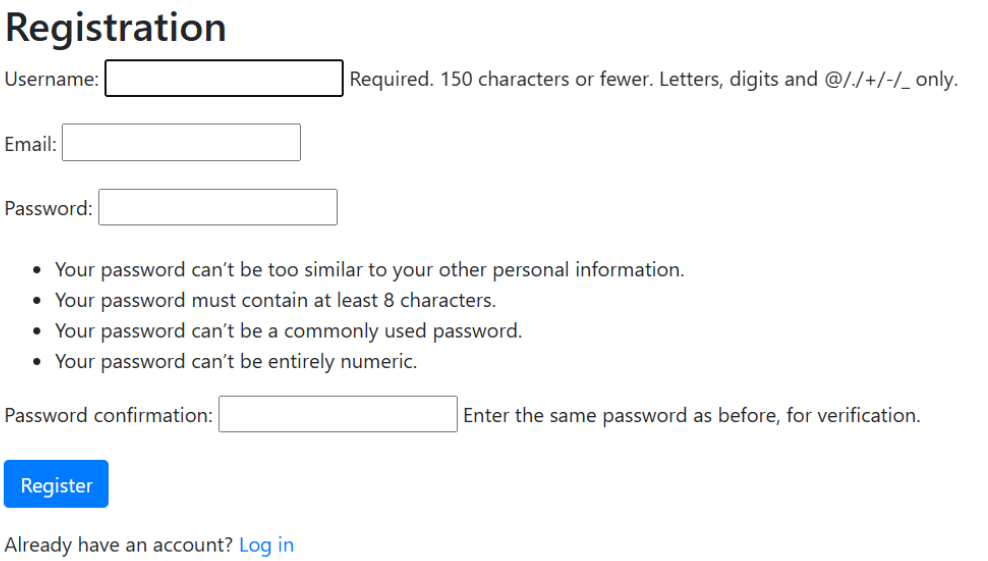

Рисунок 3.1 – Сторінка регістрації

Сторінка входу з полями «Username» і «Password», кнопкою «Log In» та посиланням «Forgot password?». Під формою — текст «Don't have an account? Register» із посиланням на форму реєстрації, а зверху — спрощене меню «Game News | Login | Register».

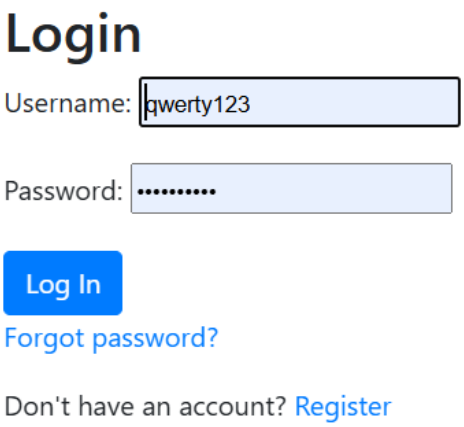

Рисунок 3.2 – Сторінка входу

Форма зміни пароля (Django PasswordChangeView) із трьома полями: старий пароль, новий пароль та підтвердження нового. Під новим паролем наведені вимоги (мінімальна довжина, складність тощо), а внизу — кнопка «Change password».

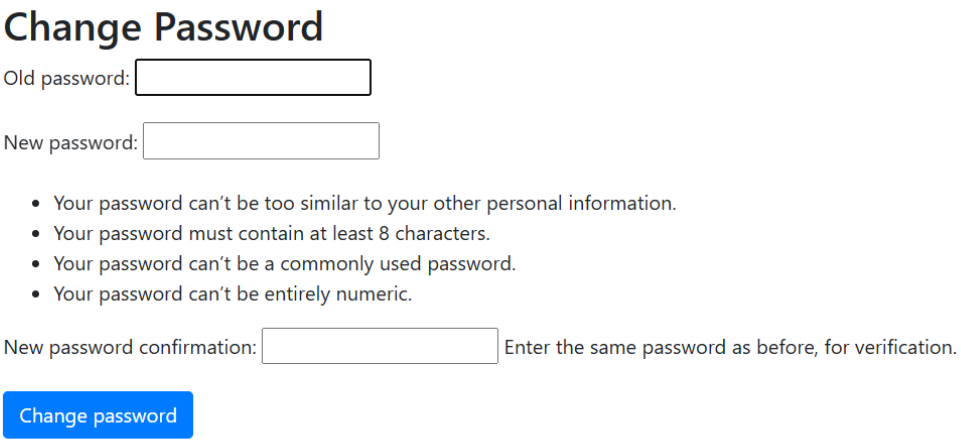

Рисунок 3.3 – Сторінка зміни паролю

Сторінка «Forgot Your Password?» із полем для введення email'у та кнопкою «Send Reset Email». Наверху — заголовок і коротка інструкція, зверху

праворуч — посилання «Login» і «Register» (оскільки користувач неавторизований).

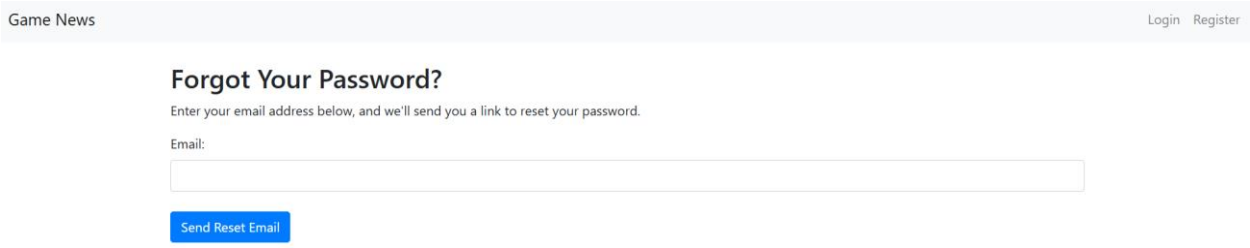

Рисунок 3.4 – Сторінка «забув пароль»

Головна сторінка «Latest News» із полем для пошуку новин, переліком категорій під ним і трьома картками з останніми статтями (зображення, категорія червоною міткою, заголовок, фрагмент тексту). Вгорі — навігаційна панель із посиланнями для авторизованого користувача.

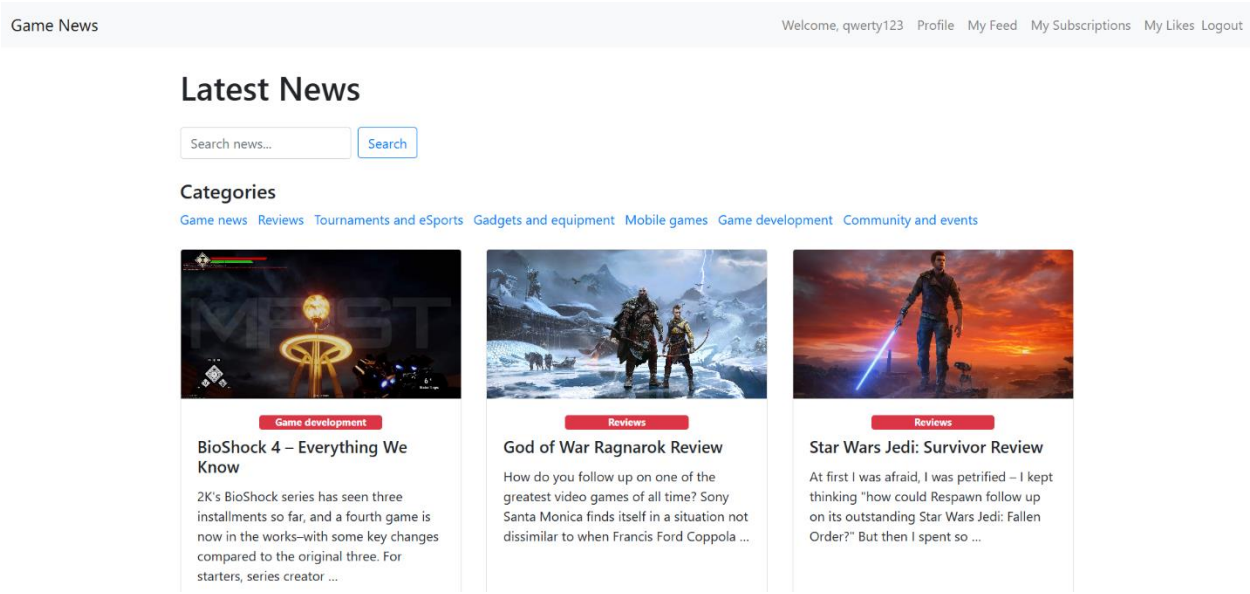

Рисунок 3.5 – Головна сторінка

Сторінка категорії “Game news” показує заголовок категорії та три картки новин із зображеннями, заголовками і коротким текстом. Угорі – кнопка

«Subscribe» та навігаційна панель із посиланнями на профіль, стрічку, підписки й лайки.

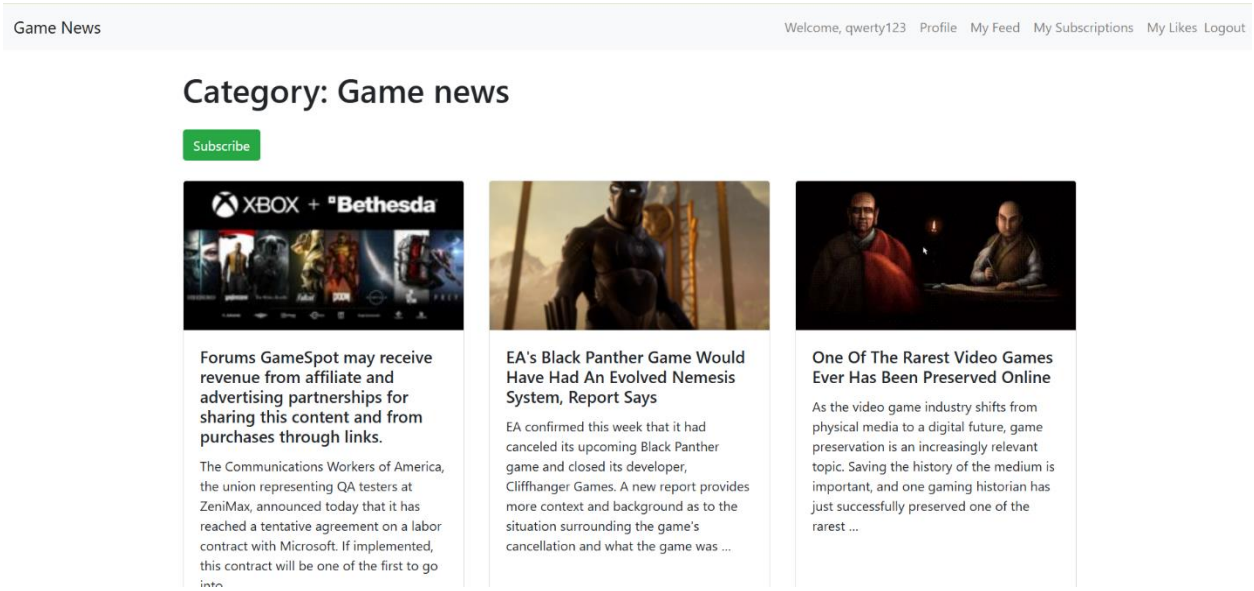

Рисунок 3.6 – Сторінка категорій новин

До речі, були реалізовані повідомлення для таких дій, як: лайки, коментарі, підписки, реєстрація.

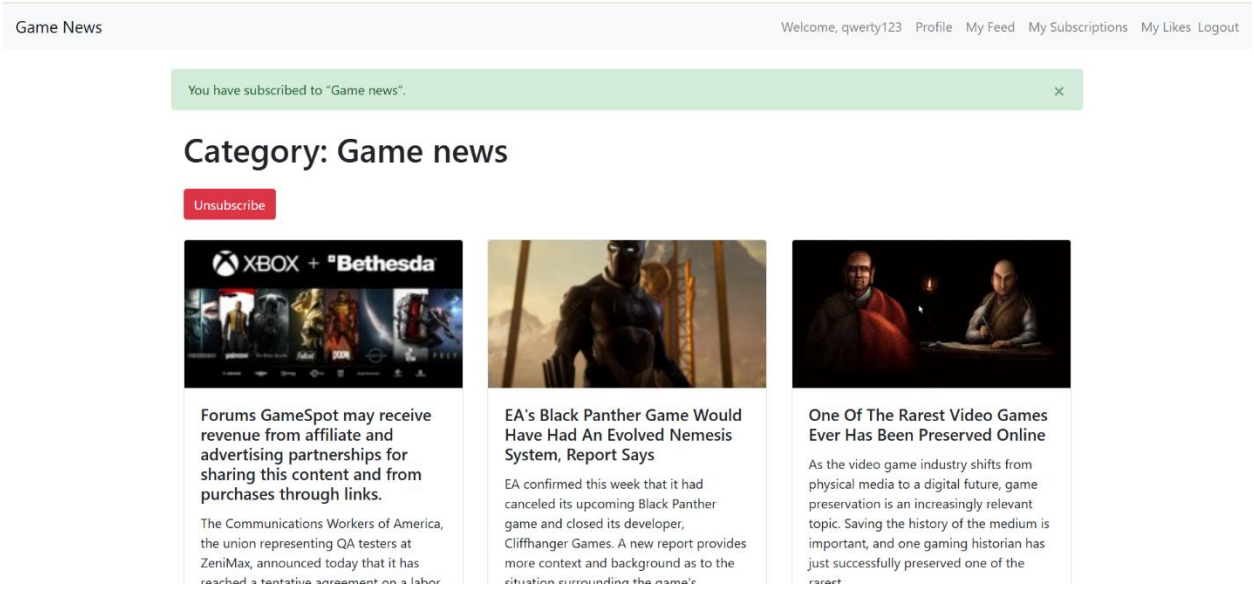

Рисунок 3.7 – демонстрація повідомлень на прикладі підписки

Сторінка «Categories You’re Subscribed To» відображає список категорій, на які користувач підписаний, та поруч з кожною — кнопку «Unsubscribe».

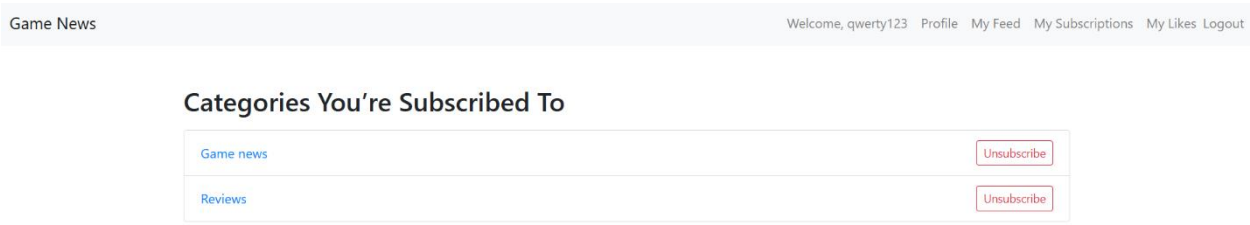

Рисунок 3.8 – Сторінка підписок користувача

Сторінка «My Feed», де відображаються картками ті новини, що належать до категорій, на які підписався користувач. Кожна картка показує зображення, категорію червоною міткою, заголовок і кнопку «Read More».

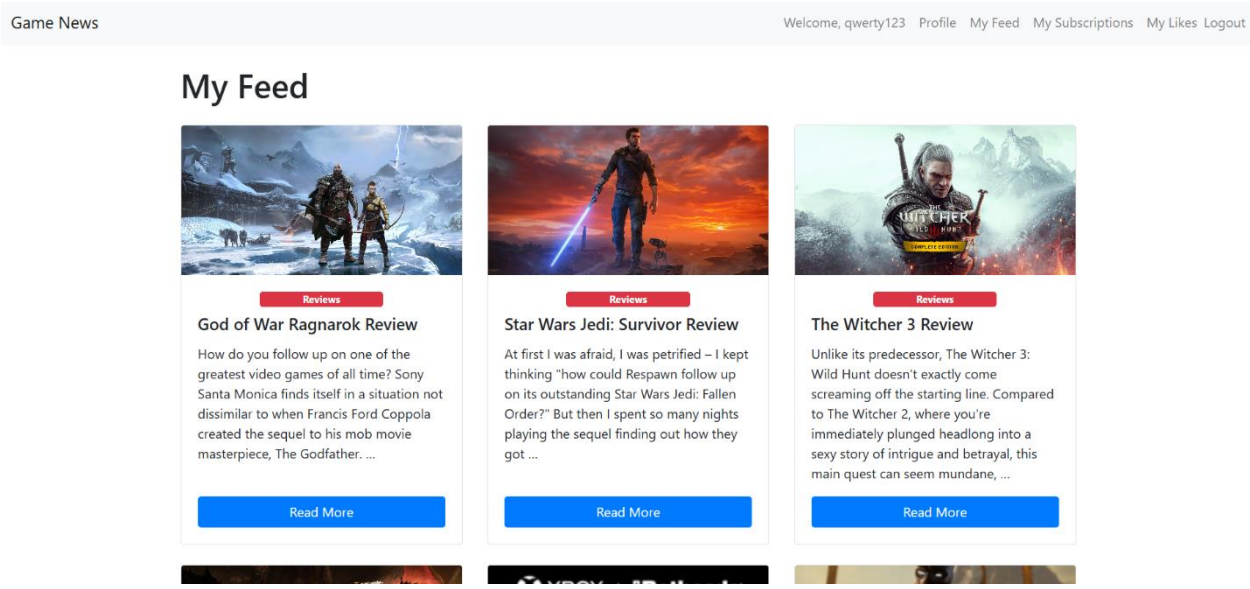

Рисунок 3.9 – Сторінка особистої стрічки новин

Сторінка із новиною. Детальний перегляд обраної статті: великий заголовок, центральне зображення й детальний опис гри нижче. У горі — навігаційна панель і заголовок сторінки.

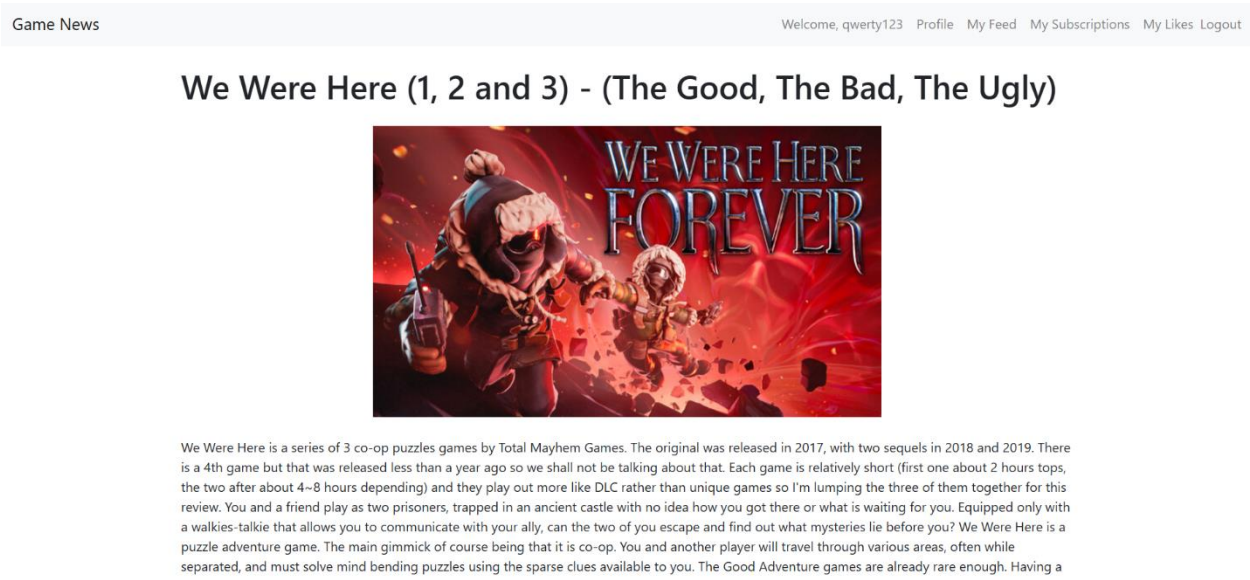

Рисунок 3.10 – Сторінка конкретної новини(статті)

На цій сторінці також є секція з коментарями. Після коментарів інших користувачів — секція «Add a Comment» із текстовим полем і кнопкою «Submit». Також можна лайкати статті та коментарі відповідно.

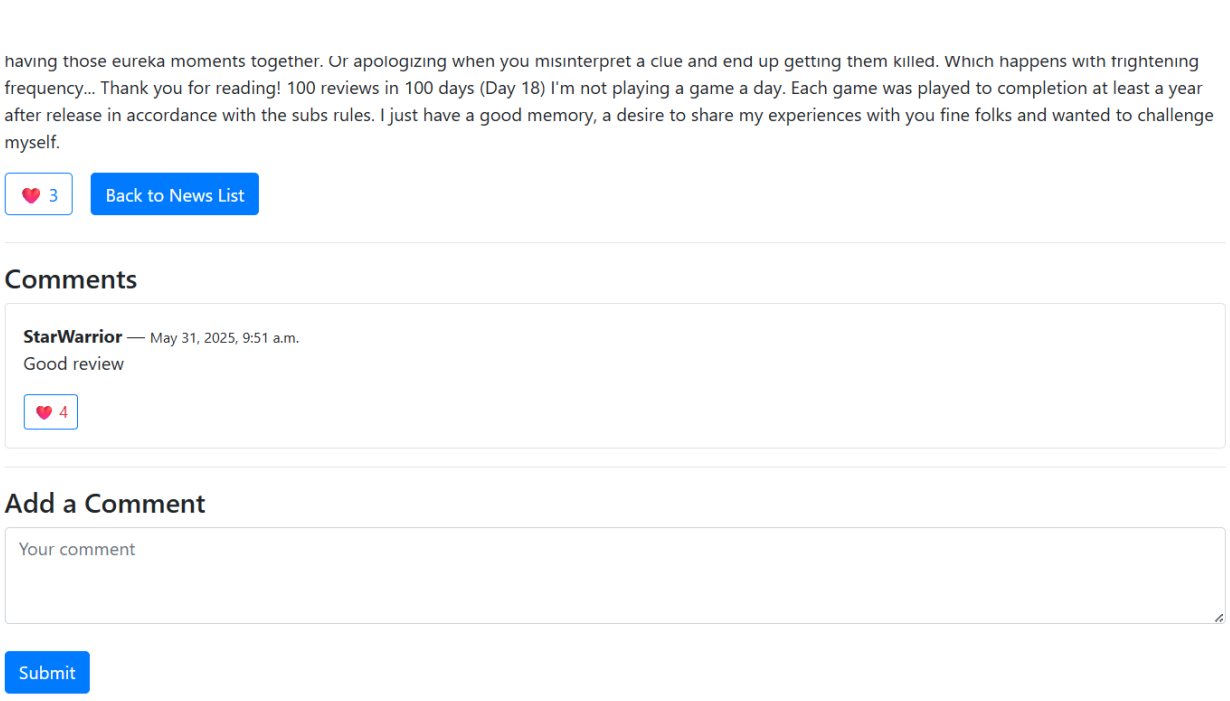

Рисунок 3.11 – Коментарі на сторінці конкретної новини(статті)

Сторінка «My Likes», розділена на два розділи: «Liked News» (список назв новин, які користувач лайкнув) та «Liked Comments» (список коментарів

із прив’язкою до статей). Кожна позиція — це посилання на відповідний контент.

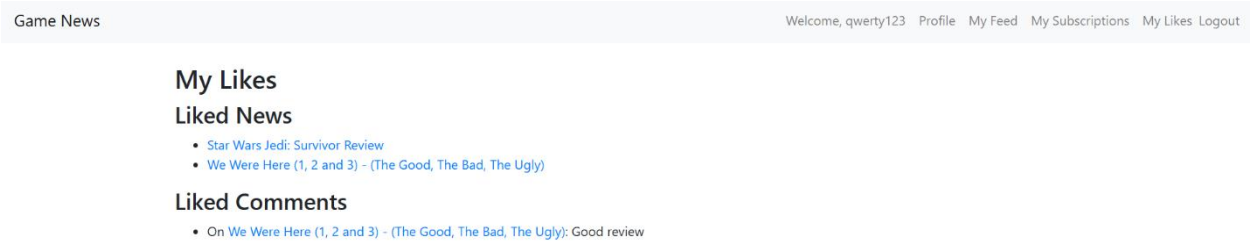

Рисунок 3.12 – Сторінка мої лайки

Сторінка «Profile» користувача, де показано їхній username і email у вигляді простих абзаців. Під цими даними — кнопка «Edit Profile» для переходу до форми редагування.

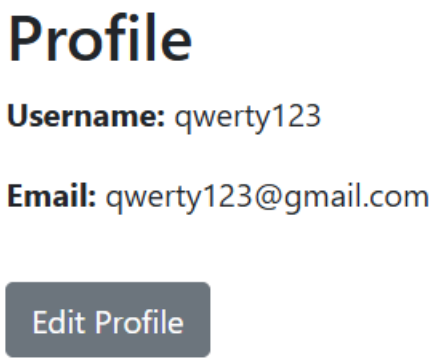

Рисунок 3.13 – Сторінка мій профіль

Форма редагування профілю користувача: поля «Username» і «Email address», уже заповнені поточними даними. Під ними кнопка «Change Password» для переходу до форми зміни пароля та кнопка «Save changes» для збереження змін.

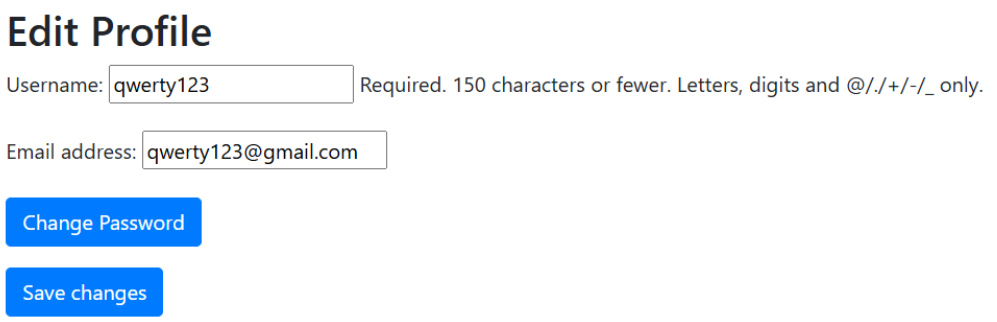

Рисунок 3.14 – Сторінка редагування профілю

Також у додатку реалізована функція пошуку статті за контентом статті(включає наповнення та назву)

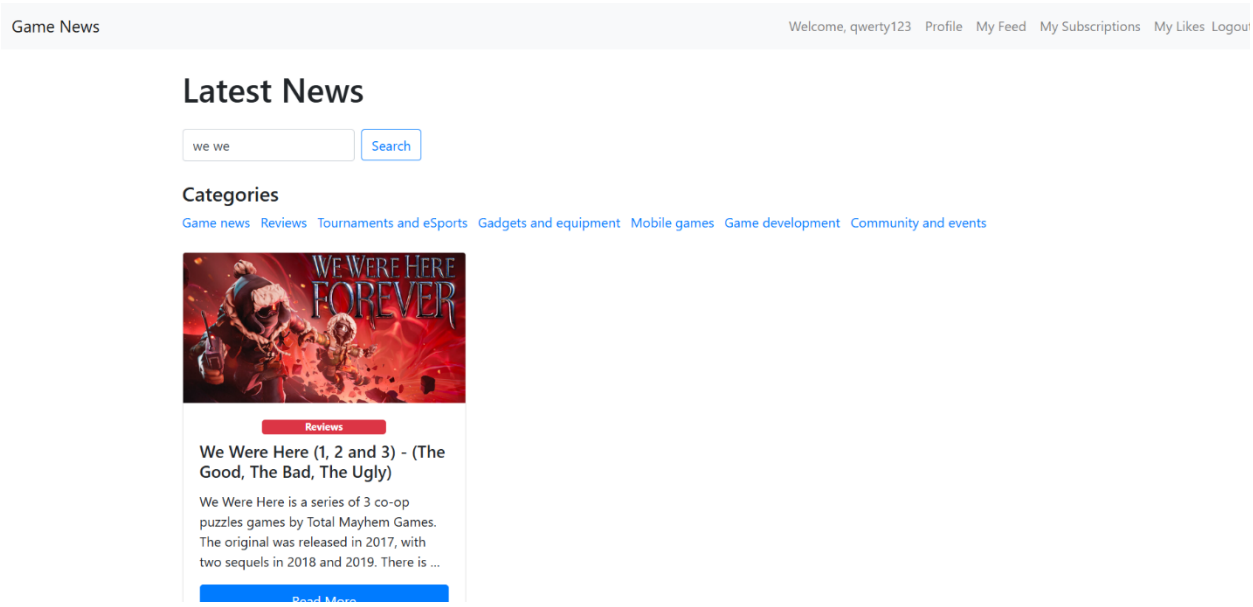

Рисунок 3.15 – Демонстрація функції пошуку

3.2. Негативні аспекти та обмеження

Використання SQLite як бази даних для розробки наклало певні обмеження на масштабування. У майбутньому планується переходити на PostgreSQL або інші системи.

Частина функціоналу, наприклад, розширені налаштування профілю чи більш складні аналітичні звіти, не реалізовані у межах поточного проекту.

Поки що відсутня мобільна версія додатку — планується розробка окремого мобільного клієнта або адаптація існуючого дизайну.

Отже, розроблений портал є надійною основою для подальшого розвитку і вдосконалення, задовольняє основні потреби цільової аудиторії та відповідає сучасним стандартам веб-розробки.

3.3. Області використання

Передбачувані області використання результатів:

- Використання як платформа для публікації геймерських новин і матеріалів.
- Основа для створення спільнот користувачів за інтересами через підписки і коментарі.
- База для впровадження реклами, партнерських програм або платних підписок.
- Навчальний матеріал для розробників, які вивчають Django та веб-технології.

Господарська, наукова та соціальна значущість:

Результати роботи мають практичну цінність для медіа-індустрії і геймерського співтовариства, сприяють покращенню доступу до актуальної інформації та підтримці інтересу користувачів. З наукової точки зору, реалізовані підходи можуть бути застосовані в інших сферах веб-розробки. Соціальна значущість полягає у створенні платформи для обміну думками і досвідом, що підтримує спільноти та формує позитивний інформаційний простір.

ВИСНОВКИ

1. Досягнення мети і завдань дослідження

В рамках виконання дипломної роботи було розроблено сучасний інтернет-портал геймерських новин, що відповідає актуальним вимогам користувачів та тенденціям веб-розробки. Метою дослідження було створення зручного, функціонального та безпечного веб-ресурсу, що дозволяє оперативно публікувати новини, забезпечувати інтерактивну взаємодію з користувачами через коментарі, лайки та підписки. В ході роботи було реалізовано всі основні компоненти системи: система користувачів з авторизацією та реєстрацією, модулі новин, коментарів, лайків і підписок, а також забезпечено адаптивність дизайну для коректного відображення на різних пристроях. Ці завдання були виконані із застосуванням найсучасніших технологій, що значно підвищує якість кінцевого продукту.

2. Особливості та цінність розробленої системи

Розробка базувалась на потужному та стабільному фреймворку Django, який забезпечив зручний інструментарій для швидкого створення надійного бекенду. Застосування модульної архітектури дозволило розділити проект на логічні частини, що полегшує як розробку, так і подальше масштабування та підтримку системи.

Використання Bootstrap для фронтенду гарантувало створення сучасного, адаптивного інтерфейсу, який відповідає сучасним стандартам UX/UI. Застосування AJAX для окремих функцій (наприклад, обробка лайків без перезавантаження сторінки) підвищує комфорт користувачів і робить взаємодію з порталом більш динамічною та приємною.

Безпека проекту є пріоритетом: застосовано всі базові механізми захисту, включаючи захист від CSRF, XSS-атак, SQL-ін’єкцій, а також реалізовано підтвердження електронної пошти користувачів для зниження ризику фальшивих акаунтів.

3. Виявлені проблеми та обмеження системи

Під час реалізації проекту було виявлено низку складнощів і обмежень, характерних для подібних систем. Зокрема, при великій кількості одночасних користувачів навантаження на сервер зростає, що вимагає додаткової оптимізації продуктивності, зокрема шляхом впровадження кешування на різних рівнях (наприклад, кешування шаблонів, результатів запитів до БД) і застосування асинхронної обробки подій.

Крім того, реалізація масштабованих механізмів управління підписками та персоналізованої стрічки новин залишається перспективною задачею для подальшого розвитку. Поточна реалізація орієнтована на базові потреби користувачів і забезпечує стабільну роботу системи на невеликих і середніх навантаженнях.

4. Перспективи подальшого розвитку та удосконалення

Потенціал розробленої платформи значний і дозволяє в майбутньому реалізувати низку важливих функцій, що підвищать її конкурентоспроможність і привабливість для користувачів. Серед можливих напрямів розвитку — впровадження багатомовності, що дозволить охопити ширшу аудиторію.

Рекомендаційні системи на основі аналізу поведінки користувачів і машинного навчання можуть значно покращити якість контенту, який вони отримують. Також доцільним є розробка мобільного додатку, що дозволить користувачам бути завжди в курсі новин.

Технічні вдосконалення включають перехід на більш потужні реляційні бази даних, інтеграцію систем кешування та використання сучасних сервісів для обробки повідомлень і подій у реальному часі. Важливим є також підвищення рівня автоматизації тестування і впровадження CI/CD для прискорення релізів.

5. Значущість роботи

Розроблений інтернет-портал геймерських новин має як практичне, так і наукове значення. З практичної точки зору, портал може бути використаний як

платформа для об’єднання геймерської спільноти, забезпечення оперативного доступу до якісної інформації, обміну думками та формування активного онлайн-середовища.

Науково-технічне значення полягає у застосуванні сучасних веб-технологій і методів розробки у контексті реального проекту, що дозволяє закріпити отримані знання, виявити і розв’язати практичні проблеми, а також запропонувати шляхи їх подальшого удосконалення.

Соціальна значущість проявляється у сприянні розвитку інтернет-комунікацій, популяризації культури відповідального споживання інформації та підвищенні рівня цифрової грамотності користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Економічна правда [Електронний ресурс] // Epravda.com.ua. – Режим доступу: https://epravda.com.ua/publications/2023/05/18/700238/ (дата звернення: 25.05.2025).
2. Django [Електронний ресурс] // Вікіпедія. – Режим доступу: https://uk.wikipedia.org/wiki/Django (дата звернення: 25.05.2025).
3. IGN [Електронний ресурс] // IGN.com. – Режим доступу: https://www.ign.com (дата звернення: 25.06.2025).
4. GameSpot [Електронний ресурс] // GameSpot.com. – Режим доступу: https://www.gamespot.com (дата звернення: 25.06.2025).
5. Kotaku [Електронний ресурс] // Kotaku.com. – Режим доступу: https://kotaku.com/ (дата звернення: 25.06.2025).
6. Як створюють інтернет портали: що це і навіщо? [Електронний ресурс] // Centum-D. – Режим доступу: https://www.centum-d.com/ru/kak-sozdayut-internet-portaly-chto-eto-i-zachem/ (дата звернення: 25.05.2025).
7. Newman S. Building Microservices: Designing Fine-Grained Systems. Sebastopol: O’Reilly Media, 2015. – 288 с (дата звернення: 26.05.2025).
8. Holovaty A., Kaplan-Moss J. The Definitive Guide to Django: Web Development Done Right. Berkeley: Apress, 2009. – 368 с (дата звернення: 26.05.2025).
9. Richardson L., Ruby S. RESTful Web Services. Sebastopol: O’Reilly Media, 2007. – 264 с (дата звернення: 26.05.2025).
10. Caching Strategies, Tools: Redis, CDN [Електронний ресурс] // Medium. – Режим доступу: https://codefarm0.medium.com/caching-strategies-tools-redis-cdn-f8f86f986c26 (дата звернення: 27.05.2025).
11. Kowert T., Quandt T. Video Games and the Gamer Generation. New York: Routledge, 2020. – 224 с (дата звернення: 27.05.2025).
12. Johnson J. Social Integration in Game Communities // Gaming Studies Journal. – 2018. – Vol. 5, No. 2. – С. 45–62 (дата звернення: 27.05.2025).

13. Mäyrä F. An Introduction to Game Studies. London: SAGE Publications, 2008. – 256 с (дата звернення: 28.05.2025).
14. Kim H. Personalization Techniques in Modern Web Portals // Journal of Web Engineering. – 2019. – Vol. 18, No. 4. – С. 273–289 (дата звернення: 28.05.2025).
15. Hamari J., Koivisto J., Sarsa H. Does Gamification Work? A Literature Review of Empirical Studies on Gamification // Proceedings of the 47th Hawaii International Conference on System Sciences. – 2014. – Розділ 8. – С. 3025–3034 (дата звернення: 28.05.2025).
16. Sullivan J. WordPress for News Portals: Best Practices // CMS Magazine. – 2019. – Vol. 12, No. 4. – С. 22–28 (дата звернення: 28.05.2025).
17. WordPress [Електронний ресурс] // WordPress.org. – Режим доступу: https://uk.wordpress.org/ (дата звернення: 27.05.2025).
18. Brown R., Johnson L. Building Scalable Web Applications with Django. Chicago: TechPress, 2020. – 312 с (дата звернення: 29.05.2025).
19. Django [Електронний ресурс] // DjangoProject.com. – Режим доступу: https://www.djangoproject.com/ (дата звернення: 13.05.2025).
20. Smith J. High-Performance Django: Optimizing Web Applications. Boston: WebDev Publishing, 2018. – 256 с (дата звернення: 22.05.2025).
21. Nguyen T., Patel R., Chen L. Single Page Applications: React, Vue.js, Angular. London: FrontEnd Books, 2021. – 184 с (дата звернення: 28.05.2025).
22. React (Legacy) [Електронний ресурс] // Reactjs.org. – Режим доступу: https://ru.legacy.reactjs.org/ (дата звернення: 29.05.2025).
23. Thompson P. Microservices and Cloud-Native Design. New York: CloudTech Press, 2022. – 240 с (дата звернення: 24.05.2025).
24. Amazon S3 [Електронний ресурс] // Amazon Web Services. – Режим доступу: https://aws.amazon.com/ru/s3/ (дата звернення: 21.05.2025).
25. Геймери [Електронний ресурс] // Вікіпедія. – Режим доступу: https://ru.wikipedia.org/wiki/Геймер (дата звернення: 21.05.2025).

26. Cybersecurity Recommendations for Protecting Personal Data [Електронний ресурс]//Newline.tech. – Режим доступу: https://newline.tech/cybersecurity-recommendations-for-protecting-personal-data-uk/ (дата звернення: 26.05.2025).
27. Bootstrap [Електронний ресурс] // getbootstrap.com. – Режим доступу: https://getbootstrap.com/ (дата звернення: 27.05.2025).
28. SQLite [Електронний ресурс] // sqlite.org. – Режим доступу: https://www.sqlite.org/ (дата звернення: 29.05.2025).
29. Django Allauth [Електронний ресурс] // Allauth.readthedocs.io. – Режим доступу: https://docs.allauth.org/en/latest/ (дата звернення: 15.05.2025).

ДОДАТОК А

БЕКЕНД

Лістинг А.1

```
from django.shortcuts import render, redirect
from .forms import UserRegisterForm
from django.contrib import messages
from django.contrib.auth.decorators import login_required
from .forms import UserUpdateForm
from django.core.paginator import Paginator
from news.models import News

def register(request):
    if request.method == 'POST':
        form = UserRegisterForm(request.POST)
        if form.is_valid():
            form.save()
            username = form.cleaned_data.get('username')
            messages.success(request, f'The account was created
for {username}!')
            return redirect('users:login')
    else:
        form = UserRegisterForm()
    return render(request, 'users/register.html', {'form':
form})

@login_required
def profile(request):
    return render(request, 'users/profile.html')

@login_required
def profile_edit(request):
    if request.method == 'POST':
        form = UserUpdateForm(request.POST,
instance=request.user)
        if form.is_valid():
            form.save()
            return redirect('users:profile')
    else:
        form = UserUpdateForm(instance=request.user)
    return render(request, 'users/profile_edit.html', {'form':
form})

@login_required
def profile_likes(request):
    user = request.user
    liked_news = user.liked_news.all()
    liked_comments = user.liked_comments.all()
    return render(request, 'users/profile_likes.html', {
        'liked_news': liked_news,
```

```
        'liked_comments': liked_comments,
    })
@login_required
def profile_subscriptions(request):
    user = request.user
    subscribed_categories = user.subscribed_categories.all()
    return render(request, 'users/profile_subscriptions.html', {
        'subscribed_categories': subscribed_categories
    })
```

Лістинг А.2

```
from django.shortcuts import render, get_object_or_404, redirect
from .models import News, Comment, Category
from django.contrib.auth.models import User
from django.db.models import Q
from django.core.paginator import Paginator
from django.contrib.auth.decorators import login_required
from django.conf import settings
from django.http import JsonResponse, HttpResponseForbidden
from .forms import CommentForm
from django.contrib import messages

# Головна сторінка новин
def index(request):
    query = request.GET.get('q')
    categories = Category.objects.all()
    if query:
        news_list = News.objects.filter(
            Q(title__icontains=query) |
Q(content__icontains=query)
        ).order_by('-created_at')
    else:
        news_list = News.objects.all().order_by('-created_at')

    paginator = Paginator(news_list, 6)  # по 6 новин на
сторінку
    page_number = request.GET.get('page')
    page_obj = paginator.get_page(page_number)

    return render(request, 'news/index.html', {
        'categories': categories,
        'page_obj': page_obj,
        'query': query,
    })


# Детальна сторінка новини
def news_detail(request, id):
    news_item = get_object_or_404(News, id=id)
    comments = news_item.comments.all().order_by('-created_at')
```

```
    if request.method == 'POST':
        if not request.user.is_authenticated:
            return
redirect(f'{settings.LOGIN_URL}?next={request.path}')
        content = request.POST.get('content')
        if content:
            Comment.objects.create(news=news_item,
author=request.user, content=content)
            return redirect('news:detail', id=news_item.id)

    return render(request, 'news/detail.html', {'news_item':
news_item, 'comments': comments})



def news_by_category(request, category_id):
    category = get_object_or_404(Category, id=category_id)
    news = category.news.all().order_by('-created_at')
    return render(request, 'news/news_by_category.html',
{'category': category, 'news': news})
```

Лістинг А.3

```
from django.shortcuts import render, get_object_or_404, redirect
from .models import News, Comment, Category
from django.contrib.auth.models import User
from django.db.models import Q
from django.core.paginator import Paginator
from django.contrib.auth.decorators import login_required
from django.conf import settings
from django.http import JsonResponse, HttpResponseForbidden
from .forms import CommentForm
from django.contrib import messages

@login_required
def edit_comment(request, comment_id):
    comment = get_object_or_404(Comment, id=comment_id,
author=request.user)
    if request.method == 'POST':
        form = CommentForm(request.POST, instance=comment)
        if form.is_valid():
            form.save()
            messages.success(request, 'Comment updated
successfully.')
            return redirect('news:detail', id=comment.news.id)
    else:
        form = CommentForm(instance=comment)
    return render(request, 'news/edit_comment.html', {'form':
form, 'comment': comment})

@login_required
def delete_comment(request, comment_id):
```

```
    comment = get_object_or_404(Comment, id=comment_id,
author=request.user)
    news_id = comment.news.id
    if request.method == 'POST':
        comment.delete()
        messages.success(request, 'Comment deleted.')
        return redirect('news:detail', id=news_id)
    return render(request, 'news/delete_comment_confirm.html',
{'comment': comment})
```

Лістинг А.4

```
from django.shortcuts import render, get_object_or_404, redirect
from .models import News, Comment, Category
from django.contrib.auth.models import User
from django.db.models import Q
from django.core.paginator import Paginator
from django.contrib.auth.decorators import login_required
from django.conf import settings
from django.http import JsonResponse, HttpResponseForbidden
from .forms import CommentForm
from django.contrib import messages

@login_required
def news_like(request, id):
    news_item = get_object_or_404(News, id=id)
    user = request.user
    if user in news_item.likes.all():
        news_item.likes.remove(user)
        liked = False
        messages.info(request, f'You unliked
“{news_item.title}.”')
    else:
        news_item.likes.add(user)
        liked = True
        messages.success(request, f'You liked
“{news_item.title}.”')

    return JsonResponse({'liked': liked, 'total_likes':
news_item.likes.count()})

@login_required
def comment_like(request, comment_id):
    comment = get_object_or_404(Comment, id=comment_id)
    user = request.user

    if user in comment.likes.all():
        comment.likes.remove(user)
        liked = False
    else:
        comment.likes.add(user)
        liked = True
```

```
    return JsonResponse({'liked': liked, 'total_likes':
comment.likes.count()})
```

Лістинг А.5

```
from django.shortcuts import render, get_object_or_404, redirect
from .models import News, Comment, Category
from django.contrib.auth.models import User
from django.db.models import Q
from django.core.paginator import Paginator
from django.contrib.auth.decorators import login_required
from django.conf import settings
from django.http import JsonResponse, HttpResponseForbidden
from .forms import CommentForm
from django.contrib import messages

def subscribe_category(request, category_id):
    category = get_object_or_404(Category, id=category_id)
    user = request.user

    if user not in category.subscribers.all():
        category.subscribers.add(user)
        messages.success(request, f'You have subscribed to
“{category.name}”.')
    else:
        messages.info(request, f'You are already subscribed to
“{category.name}”.')

    return redirect('news:news_by_category',
category_id=category_id)

@login_required
def unsubscribe_category(request, category_id):
    category = get_object_or_404(Category, id=category_id)
    user = request.user

    if user in category.subscribers.all():
        category.subscribers.remove(user)
        messages.success(request, f'You have unsubscribed from
“{category.name}”.')
    else:
        messages.info(request, f'You were not subscribed to
“{category.name}”.')

    return redirect('news:news_by_category',
category_id=category_id)

@login_required
def profile_subscriptions(request):
    user = request.user
    subscribed_categories = user.subscribed_categories.all()
    return render(request, 'users/profile_subscriptions.html', {
```

```
        'subscribed_categories': subscribed_categories
    })
@login_required
def my_feed(request):
    user = request.user
    # Беремо всі новини, категорії яких мають цього користувача
в subscribers
    feed_qs =
News.objects.filter(category__subscribers=user).order_by('-
created_at')

    # Пагінація: 9 записів на сторінку (можна змінити)
    paginator = Paginator(feed_qs, 9)
    page_number = request.GET.get('page')
    page_obj = paginator.get_page(page_number)

    return render(request, 'users/my_feed.html', {
        'page_obj': page_obj,
    })
```

ДОДАТОК Б

ФРОНТЕНД

Лістинг Б.1

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-
scale=1" />
  <title>{% block title %}Game News{% endblock %}</title>

  <!-- Bootstrap CSS -->
  <link
    rel="stylesheet"

href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/boo
tstrap.min.css"
  />
  <!-- (за бажанням) Font Awesome для іконок -->
  <link
    rel="stylesheet"
    href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css"
  />
</head>
<body>
  <!-- ПОЧАТОК ВЕРХНЬОЇ НАВІГАЦІЙНОЇ ПАНЕЛІ -->
  <nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand" href="{% url 'news:index' %}">Game
News</a>
    <button
      class="navbar-toggler"
      type="button"
      data-toggle="collapse"
      data-target="#navbarNav"
      aria-controls="navbarNav"
      aria-expanded="false"
      aria-label="Toggle navigation"
    >
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
      <ul class="navbar-nav ml-auto">
        {% if user.is_authenticated %}
          <li class="nav-item">
            <span class="nav-link">Welcome, {{ user.username
}}</span>
          </li>
          <li class="nav-item">
```

```
            <a class="nav-link" href="{% url 'users:profile'
%}">Profile</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{% url 'users:my_feed'
%}">My Feed</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{% url
'users:profile_subscriptions' %}">My Subscriptions</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{% url
'users:profile_likes' %}">My Likes</a>
          </li>
          <li class="nav-item">
            <form method="post" action="{% url 'users:logout'
%}" style="display: inline;">
              {% csrf_token %}
              <button
                type="submit"
                class="nav-link btn btn-link"
                style="
                  padding: 0;
                  margin: 0;
                  border: none;
                  background: none;
                  vertical-align: middle;
                  line-height: 2.5;
                "
              >
                Logout
              </button>
            </form>
          </li>
        {% else %}
          <li class="nav-item">
            <a class="nav-link" href="{% url 'users:login'
%}">Login</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{% url 'users:register'
%}">Register</a>
          </li>
        {% endif %}
      </ul>
    </div>
  </nav>
  <!-- КІНЕЦЬ ВЕРХНЬОЇ НАВІГАЦІЙНОЇ ПАНЕЛІ -->

  <!-- FLASH MESSAGES -->
  {% if messages %}
```

```
    <div class="container mt-4">
      {% for message in messages %}
        <div class="alert
          {% if message.tags %}alert-{{ message.tags }}{% else
%}alert-info{% endif %}
          alert-dismissible fade show" role="alert">
          {{ message }}
          <button type="button" class="close" data-
dismiss="alert" aria-label="Close">
            <span aria-hidden="true">&times;</span>
          </button>
        </div>
      {% endfor %}
    </div>
  {% endif %}

  <!-- ОСНОВНИЙ КОНТЕНТ -->
  <div class="container mt-4">
    {% block content %}
    {% endblock %}
  </div>

  <!-- FOOTER -->
  <footer class="bg-light text-center text-lg-start mt-5 border-
top">
    <div class="container p-4">
      <div class="row">
        <!-- Коротка інформація -->
        <div class="col-lg-4 col-md-6 mb-4 mb-md-0">
          <h5 class="text-uppercase">Game News</h5>
          <p class="text-muted">
            Your daily dose of gaming updates: reviews,
trailers, interviews and more.
          </p>
        </div>
        <!-- Швидкі посилання -->
        <div class="col-lg-4 col-md-6 mb-4 mb-md-0">
          <h5 class="text-uppercase">Quick Links</h5>
          <ul class="list-unstyled mb-0">
            <li><a href="{% url 'news:index' %}" class="text-
dark">Home</a></li>
            <li><a href="{% url 'news:index' %}?q=" class="text-
dark">Search</a></li>
            <li><a href="{% url 'users:my_feed' %}" class="text-
dark">My Feed</a></li>
            <li><a href="{% url 'users:profile_subscriptions'
%}" class="text-dark">Subscriptions</a></li>
          </ul>
        </div>
        <!-- Контакти -->
        <div class="col-lg-4 col-md-12 mb-4 mb-md-0">
          <h5 class="text-uppercase">Contact</h5>
```

```
          <ul class="list-unstyled mb-0">
            <li><span class="text-dark">Email:
support@gamenews.example</span></li>
            <li><span class="text-dark">Phone: +1 234 567
890</span></li>
            <li class="mt-2">
              <a href="#" class="text-dark mr-3"><i class="fab
fa-facebook"></i></a>
              <a href="#" class="text-dark mr-3"><i class="fab
fa-twitter"></i></a>
              <a href="#" class="text-dark"><i class="fab fa-
instagram"></i></a>
            </li>
          </ul>
        </div>
      </div>
    </div>
    <div class="text-center p-3 bg-dark text-white">
      © 2025 Game News. All rights reserved.
    </div>
  </footer>

  <!-- Bootstrap JS (для роботи кнопок і алертів) -->
  <script src="https://code.jquery.com/jquery-
3.5.1.slim.min.js"></script>
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.2/dist/js/bootst
rap.bundle.min.js"></script>
</body>
</html>
```

Лістинг Б.2

```
{% extends 'base.html' %}

{% block title %}Latest News{% endblock %}

{% block content %}
<div class="container">
  <h1 class="my-4">Latest News</h1>

  <!-- Search form -->
  <form method="get" action="{% url 'news:index' %}"
class="form-inline mb-4">
    <input
      type="text"
      name="q"
      class="form-control mr-2"
      placeholder="Search news..."
      value="{{ request.GET.q }}"
    />
```

```
    <button type="submit" class="btn btn-outline-
primary">Search</button>
  </form>

  <!-- Categories -->
  <div class="mb-4">
    <h4>Categories</h4>
    <ul class="list-inline">
      {% for category in categories %}
        <li class="list-inline-item">
          <a href="{% url 'news:news_by_category' category.id
%}">{{ category.name }}</a>
        </li>
      {% empty %}
        <li>No categories available.</li>
      {% endfor %}
    </ul>
  </div>

  <!-- News list -->
  <div class="row">
    {% for news_item in page_obj %}
      <div class="col-md-4 mb-4">
        <div class="card h-100">
          {% if news_item.image %}
            <div class="card-img-top" style="overflow: hidden;
height: 180px;">
              <img
                src="{{ news_item.image.url }}"
                class="img-fluid w-100"
                style="object-fit: cover; height: 100%;"
                alt="{{ news_item.title }}"
              />
            </div>
          {% endif %}

          <div class="card-body d-flex flex-column">
            <!-- Червона мітка категорії над заголовком, шириною
50% -->
            <span class="badge badge-danger w-50 mb-2 text-
center mx-auto">
              {{ news_item.category.name }}
            </span>

            <h5 class="card-title">{{ news_item.title }}</h5>
            <p class="card-text">{{
news_item.content|truncatewords:30 }}</p>
            <a href="{% url 'news:detail' news_item.id %}"
class="btn btn-primary mt-auto">
              Read More
            </a>
          </div>
```

```
        </div>
      </div>
    {% empty %}
      <p>No news available.</p>
    {% endfor %}
  </div>

  <!-- Pagination -->
  <nav aria-label="Page navigation">
    <ul class="pagination justify-content-center">
      {% if page_obj.has_previous %}
        <li class="page-item">
          <a
            class="page-link"
            href="?{% if query %}q={{ query }}&{% endif
%}page={{ page_obj.previous_page_number }}"
            aria-label="Previous"
            >&laquo;</a
          >
        </li>
      {% else %}
        <li class="page-item disabled">
          <span class="page-link" aria-
label="Previous">&laquo;</span>
        </li>
      {% endif %}

      {% for num in page_obj.paginator.page_range %}
        {% if page_obj.number == num %}
          <li class="page-item active"><span class="page-
link">{{ num }}</span></li>
        {% else %}
          <li class="page-item">
            <a
              class="page-link"
              href="?{% if query %}q={{ query }}&{% endif
%}page={{ num }}"
              >{{ num }}</a
            >
          </li>
        {% endif %}
      {% endfor %}

      {% if page_obj.has_next %}
        <li class="page-item">
          <a
            class="page-link"
            href="?{% if query %}q={{ query }}&{% endif
%}page={{ page_obj.next_page_number }}"
            aria-label="Next"
            >&raquo;</a
          >
```

```
        </li>
      {% else %}
        <li class="page-item disabled">
          <span class="page-link" aria-
label="Next">&raquo;</span>
        </li>
      {% endif %}
    </ul>
  </nav>
</div>
{% endblock %}
```

Лістинг Б.3

```
{% extends 'base.html' %}

{% block title %}{{ news_item.title }}{% endblock %}

{% block content %}
<div class="container">
    <h1 class="my-4">{{ news_item.title }}</h1>

   {% if news_item.image %}
  <img
    src="{{ news_item.image.url }}"
    class="img-fluid mb-4 mx-auto d-block"
    alt="{{ news_item.title }}"
  />
{% endif %}


    <p>{{ news_item.content }}</p>

    <!-- Кнопка лайка новини -->
    <div class="d-flex align-items-center mb-4">
  <button id="like-btn" class="btn btn-outline-primary mr-3">
      ❤️ <span id="like-count">{{ news_item.likes.count
}}</span>
  </button>
  <a href="{% url 'news:index' %}" class="btn btn-primary">
      Back to News List
  </a>
</div>


    <hr />

    <!-- Comments section -->
    <h4>Comments</h4>
    {% if comments %}
        {% for comment in comments %}
            <div class="mb-3 p-3 border rounded">
```

```
                <strong>{{ comment.author.username }}</strong> —
<small>{{ comment.created_at }}</small>
                <p>{{ comment.content }}</p>
                <!-- Кнопка лайка коментаря -->
                <button class="btn btn-outline-primary btn-sm
comment-like-btn {% if user in comment.likes.all %}text-danger{%
endif %}" data-comment-id="{{ comment.id }}">
                    ❤️ <span class="like-count">{{
comment.likes.count }}</span>
                </button>
            {% if user == comment.author %}
      <a href="{% url 'news:edit_comment' comment.id %}"
class="btn btn-sm btn-outline-secondary">Edit</a>
      <a href="{% url 'news:delete_comment' comment.id %}"
class="btn btn-sm btn-outline-danger">Delete</a>
    {% endif %}
            </div>
        {% endfor %}
    {% else %}
        <p>No comments yet. Be the first to comment!</p>
    {% endif %}

    <hr />

    <!-- Form to add a new comment -->
    <h4>Add a Comment</h4>
    <form method="post">
        {% csrf_token %}
        <div class="form-group">
            <textarea
                name="content"
                rows="3"
                class="form-control"
                placeholder="Your comment"
                required
            ></textarea>
        </div>
        <button type="submit" class="btn btn-primary mt-
2">Submit</button>
    </form>
</div>

<script>
// Лайк для новини
document.getElementById('like-btn').addEventListener('click',
function() {
    fetch("{% url 'news:news_like' news_item.id %}", {
        method: 'POST',
        headers: {
            'X-CSRFToken': '{{ csrf_token }}',
            'Accept': 'application/json',
            'Content-Type': 'application/json'
```

```
        },
        credentials: 'same-origin',
    })
    .then(response => response.json())
    .then(data => {
        document.getElementById('like-count').textContent =
data.total_likes;
        this.classList.toggle('btn-primary', data.liked);
        this.classList.toggle('btn-outline-primary',
!data.liked);
    });
});

// Лайк для коментарів
document.querySelectorAll('.comment-like-btn').forEach(btn => {
  btn.addEventListener('click', function() {
    const commentId = this.dataset.commentId;
    fetch("{% url 'news:comment_like' 0 %}".replace('0',
commentId), {
      method: 'POST',
      headers: {
        'X-CSRFToken': '{{ csrf_token }}',
        'Accept': 'application/json',
        'Content-Type': 'application/json'
      },
      credentials: 'same-origin',
    })
    .then(res => res.json())
    .then(data => {
      this.querySelector('.like-count').textContent =
data.total_likes;
      if (data.liked) {
        this.classList.add('text-danger');
      } else {
        this.classList.remove('text-danger');
      }
    });
  });
});

</script>
{% endblock %}
```

Лістинг Б.4

```
{% extends 'base.html' %}

{% block title %}User Profile{% endblock %}

{% block content %}
<h2>Profile</h2>
<p><strong>Username:</strong> {{ request.user.username }}</p>
<p><strong>Email:</strong> {{ request.user.email }}</p>
```

```
<a href="{% url 'users:profile_edit' %}" class="btn btn-
secondary mt-3">Edit Profile</a>
{% endblock %}
```

Лістинг Б.5

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-
scale=1" />
    <title>Login</title>
    <link
      rel="stylesheet"

href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/boo
tstrap.min.css"
    />
</head>
<body>
    <div class="container mt-5">
        <h2>Login</h2>
        <form method="post">
            {% csrf_token %}
            {{ form.as_p }}
            <button type="submit" class="btn btn-primary">Log
In</button>
        </form>
        <p>
    <a href="{% url 'users:password_reset' %}">Forgot
password?</a>
</p>

        <p class="mt-3">
            Don't have an account? <a href="{% url
'users:register' %}">Register</a>
        </p>
    </div>
</body>
</html>
```

ДОДАТОК В

ТЕСТУВАННЯ

Лістинг В.1

```
from django.test import TestCase, Client
from django.urls import reverse
from django.contrib.auth.models import User
from django.utils import timezone
from news.models import News, Comment

import json

class NewsAppTestCase(TestCase):
    def setUp(self):
        # Створимо двох користувачів: одного автором новини,
іншого — звичайним
        self.author =
User.objects.create_user(username='author',
email='author@example.com', password='pass12345')
        self.other = User.objects.create_user(username='other',
email='other@example.com', password='pass12345')

        # Створимо одну новину, прив’язану до self.author
        self.news = News.objects.create(
            title='Test News',
            content='Content of the test news.',
            author=self.author,
            created_at=timezone.now()
        )

        self.client = Client()

    def test_only_authenticated_can_comment(self):
        """
        При GET детального перегляду новини повертається 200.
        При POST (додавання коментаря) без авторизації —
редирект на логін.
        Аутентифікований користувач після POST успішно створює
коментар.
        """
        detail_url = reverse('news:detail', args=[self.news.pk])

        # GET запит без авторизації — просто рендер сторінки
        response = self.client.get(detail_url)
        self.assertEqual(response.status_code, 200)

        # POST без логіну — очікуємо redirect (на логін)
        response = self.client.post(detail_url, {'content':
'Anonymous comment'})
        self.assertEqual(response.status_code, 302)
        self.assertIn('/users/login/', response['Location'])
```

```
        # Тепер залогінимося іншим користувачем і спробуємо ще
раз
        self.client.login(username='other',
password='pass12345')
        response = self.client.post(detail_url, {'content':
'Authenticated comment'}, follow=True)
        self.assertEqual(response.status_code, 200)

        # Перевіримо, що у базі з’явився коментар
        comment_qs = Comment.objects.filter(news=self.news,
author=self.other, content='Authenticated comment')
        self.assertTrue(comment_qs.exists())
```

Лістинг В.2

```
from django.test import TestCase, Client
from django.urls import reverse
from django.contrib.auth.models import User
from django.utils import timezone
from news.models import News, Comment

import json
def test_like_toggle_ajax_updates_database(self):
        """
        Тест лайк/дизлайк для новини через AJAX-view:
        - Авторизуємося
        - Відправляємо POST на AJAX-endpoint
        - Перевіряємо JSON-відповідь та стан у базі
        - Повторне натискання знімає лайк
        """
        like_url = reverse('news:news_like', kwargs={'pk':
self.news.pk})

        # Спроба без логіну — редирект
        response = self.client.post(like_url,
HTTP_X_REQUESTED_WITH='XMLHttpRequest')
        self.assertEqual(response.status_code, 302)

        # Логінимося іншим користувачем
        self.client.login(username='other',
password='pass12345')

        # Початково у жодного лайку немає
        self.assertEqual(self.news.likes.count(), 0)

        # Відправляємо AJAX-запит на лайк
        response = self.client.post(
            like_url,
            HTTP_X_REQUESTED_WITH='XMLHttpRequest',
        )
        # Перевіримо, що вернувся JSON зі status 200
```

```
        self.assertEqual(response.status_code, 200)
        data = json.loads(response.content)
        self.assertTrue(data.get('liked'))
        self.assertEqual(data.get('total_likes'), 1)

        # У базі має бути один запис лайку (припускаємо, що у
моделі News є ManyToManyField 'likes')
        self.news.refresh_from_db()
        self.assertEqual(self.news.likes.count(), 1)

self.assertTrue(self.news.likes.filter(pk=self.other.pk).exists(
))

        # Знову натискаємо — це мусить зняти лайк
        response = self.client.post(
            like_url,
            HTTP_X_REQUESTED_WITH='XMLHttpRequest',
        )
        self.assertEqual(response.status_code, 200)
        data = json.loads(response.content)
        self.assertFalse(data.get('liked'))
        self.assertEqual(data.get('total_likes'), 0)

        self.news.refresh_from_db()
        self.assertEqual(self.news.likes.count(), 0)
```

Лістинг В.3

```
from django.test import TestCase, Client, override_settings
from django.urls import reverse
from django.contrib.auth.models import User
from django.core import mail

class UsersAppTestCase(TestCase):
    def setUp(self):
        self.client = Client()
        # Створимо користувача для тесту зміни пароля
        self.user =
User.objects.create_user(username='testuser',
email='test@example.com', password='oldpassword123')

    @override_settings(ACCOUNT_EMAIL_VERIFICATION='mandatory',
EMAIL_BACKEND='django.core.mail.backends.locmem.EmailBackend')
    def test_registration_with_email_confirmation(self):
        """
        Перевіряємо реєстрацію нового користувача з allauth:
        - Надсилаємо дані на signup
        - Переконуємося, що в базі створено непривірений акаунт
(is_active=False або allauth-specific)
        - Переконуємося, що в outbox є лист для підтвердження
email
        - Імітуємо натискання на посилання підтвердження та
перевіряємо, що акаунт активувався
```

```
        """
        signup_url = reverse('account_signup')

        data = {
            'username': 'newuser',
            'email': 'newuser@example.com',
            'password1': 'ComplexPwd!123',
            'password2': 'ComplexPwd!123',
        }
        response = self.client.post(signup_url, data)
        # Після реєстрації allauth зазвичай перенаправляє на
підтвердження email (302)
        self.assertEqual(response.status_code, 302)

        # Перевіримо, що акаунт створено, але неактивований:
allauth встановлює is_active = False поки не підтверджено
        new_user =
User.objects.filter(username='newuser').first()
        self.assertIsNotNone(new_user)
        self.assertFalse(new_user.is_active)

        # Перевірка, що був відправлений лист для підтвердження
        self.assertEqual(len(mail.outbox), 1)
        confirmation_mail = mail.outbox[0]
        self.assertIn('newuser@example.com',
confirmation_mail.to)

        # Витягнемо лінк підтвердження з тексту листа
        # allauth за замовчуванням вставляє щось на кшталт:
/accounts/confirm-email/<key>/
        url_start =
confirmation_mail.body.find('/accounts/confirm-email/')
        self.assertTrue(url_start > 0)
        # обріжемо до кінця рядка
        url_end = confirmation_mail.body.find('\n', url_start)
        confirm_path =
confirmation_mail.body[url_start:url_end].strip()
        confirm_url = confirm_path  # це шлях, починаючи з /

        # Інший спосіб: скористатися методом
EmailConfirmation.objects.get().key, але нижче - читаємо з тіла
листа

        # Симулюємо GET на confirm_url
        response = self.client.get(confirm_url, follow=True)
        self.assertEqual(response.status_code, 200)

        # Оновимо дані користувача з бази
        new_user.refresh_from_db()
        self.assertTrue(new_user.is_active)
```

Лістинг В.4

```
from django.test import TestCase, Client, override_settings
from django.urls import reverse
from django.contrib.auth.models import User
from django.core import mail

def test_password_change_checks_old_password(self):
        """
        Перевіряємо зміну пароля:
        - Анонім спробує зайти на password_change (отримає
редирект на логін)
        - Авторизований користувач, що введе неправильний
'old_password', отримає помилку форми.
        - При правильному старому паролі пароль зміниться
(редирект).
        """
        change_password_url = reverse('users:password_change')
        login_url = reverse('users:login')

        # 1) Анонім: GET -> редирект на логін
        response = self.client.get(change_password_url)
        self.assertEqual(response.status_code, 302)
        self.assertIn(login_url, response['Location'])

        # 2) Залогіньмося
        self.client.login(username='testuser',
password='oldpassword123')

        # GET запит для форми зміни пароля повертає 200
        response = self.client.get(change_password_url)
        self.assertEqual(response.status_code, 200)

        # 3) POST із невірним старим паролем
        response = self.client.post(change_password_url, {
            'old_password': 'wrongoldpwd',
            'new_password1': 'NewStrongPwd!1',
            'new_password2': 'NewStrongPwd!1',
        })
        # При невірному old_password система рендерить ту ж
сторінку з помилкою форми
        self.assertEqual(response.status_code, 200)
        self.assertFormError(response, 'form', 'old_password',
'Your old password was entered incorrectly. Please enter it
again.')

        # 4) POST із правильним старим паролем, але нові паролі
не збігаються => error
        response = self.client.post(change_password_url, {
            'old_password': 'oldpassword123',
            'new_password1': 'NewPwd!123',
            'new_password2': 'Mismatch123',
        })
```

```
        self.assertEqual(response.status_code, 200)
        # Має бути помилка 'new_password2'
        self.assertFormError(response, 'form', 'new_password2',
"The two password fields didn’t match.")

        # 5) POST із правильними старим і новими паролями
        response = self.client.post(change_password_url, {
            'old_password': 'oldpassword123',
            'new_password1': 'NewPwd!123',
            'new_password2': 'NewPwd!123',
        }, follow=True)
        # Після успішної зміни зазвичай відбувається редирект
(HTTP 302), а потім отримуємо статус 200
        self.assertEqual(response.status_code, 200)

        # Перевіримо, що тепер можна заходити з новим паролем, а
зі старим — ні
        self.client.logout()

        login = self.client.login(username='testuser',
password='oldpassword123')
        self.assertFalse(login)

        login = self.client.login(username='testuser',
password='NewPwd!123')
        self.assertTrue(login)
```