МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 «Комп'ютерні науки»

освітньо-професійної програми «Інформаційні технології проектування»

на тему: Веб-орієнтована система новинного блогу

Здобувача групи ІТ.м-32 Іваненка Дмитра Ігоровича
(шифр групи) (прізвище, ім'я, по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ _____ Дмитро Іваненко _____
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник доц. кафедри, к.т.н., доц. Світлана ВАШЕНКО _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри ІТ

Світлана ВАЩЕНКО
«_____» _____ 2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Іваненку Дмитру Ігоровичу
(прізвище, ім'я, по батькові)

1 Тема кваліфікаційної роботи Веб-орієнтована система новинного блогу

затверджена наказом по університету від «11» жовтня 2024 р. № 1044-VI (зі змінами від 20 листопада 2024 р. № 1200-VI)

2 Термін здачі студентом кваліфікаційної роботи «6» грудня 2024 р.

3 Вхідні дані до кваліфікаційної роботи технічне завдання на розробку веб-орієнтованої системи підтримки діяльності новинного блогу

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) Аналіз предметної області, постановка задачі та методи дослідження, моделювання та проектування веб-орієнтованої системи, програмна реалізація.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових слайдів презентації)
вікно авторизації, вікно реєстрації, вікно окремо відкритої статті, вікно вибору і підтвердження фото, вікно списку всіх статей, вікно списку статей за категоріями, вікно списку популярних статей, вікно списку нещодавно доданих статей, вікно панелі адміністратора.

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
	Дослідження предметної області	30.08.2024 – 17.10.2024	
	Розробка веб додатку	17.10.2024 – 14.11.2024	
	Тестування	14.11.2024 – 26.11.2024	
	Перевірка працездатності	26.11.2024 – 28.11.2024	
	Створення програмної документації	28.11.2024 – 05.12.2024	

Магістрант _____ Дмитро ІВАНЕНКО

Керівник роботи _____ к.т.н., доц. Світлана ВАЩЕНКО

АНОТАЦІЯ

Тема кваліфікаційної роботи магістра «Веб орієнтована система новинного блогу».

Пояснювальна записка складається зі вступу, 4 розділів, висновку, списку використаних джерел із 35 найменувань, додатків. Загальний обсяг роботи – 111 сторінок, у тому числі 63 сторінок основного тексту, 5 сторінок списку використаних джерел, 33 сторінки додатків.

Кваліфікаційну роботу магістра присвячено розробці веб орієнтованої системи підтримки діяльності новинного блогу.

В роботі проведено аналіз предметної області, оцінка вже подібних існуючих додатків. Досліджено засоби реалізації основного функціоналу системи. Спроектовано та розроблено веб додаток.

У роботі виконано огляд останніх досліджень та публікацій, аналіз програмних продуктів-аналогів, функціональне моделювання програмного додатку в IDEF0, моделювання варіантів використання та проектування бази даних.

У роботі було продемонстровано архітектуру веб додатку, реалізацію бази даних, програмну реалізацію та використання веб додатку.

Результатом проведеної роботи є створення веб орієнтованої системи підтримки діяльності новинного блогу.

Практичне значення роботи полягає у розробці веб орієнтованої системи новинного блогу.

Ключові слова: веб додаток, блог, новини, база даних, yii2, управління процесами, framework, web, MySQL

ЗМІСТ

ВСТУП

Враховуючи швидкий розвиток інформаційних технологій та зростаючу роль цифрових медіа в сучасному суспільстві, з’являється нагальна потреба в створенні та розробці інтуїтивно зрозумілих і функціональних веб-додатків. Особливо це актуально для новинних блогів, які повинні відповідати вимогам користувачів і здатними забезпечити новітні підходи до управління контентом та взаємодії з аудиторією. Актуальність даної теми обумовлена потребою в удосконаленні роботи новинних порталів, що може сприяти підвищенню їх популярності та позитивно впливати на загальний інформаційний ландшафт країни.

Ключовими завданнями, які постають перед розробниками веб-додатків такого типу, є забезпечення безпеки користувацьких даних, інтеграція механізмів авторизації та ефективне управління контентом, а також створення можливостей для взаємодії користувачів через коментарі та інші інтерактивні елементи. З огляду на ці аспекти та вимоги, дослідження в цій сфері є актуальним, оскільки це дозволить розробити веб орієнтовану систему, яка задовольнить потребу сучасних користувачів та забезпечить надійність і зручність використання новинного блогу.

Об'єктом дослідження є функціональні особливості веб додатків новинних платформ та їх архітектури.

Предметом дослідження є веб-додаток для новинних платформ, методи розробки інтерфейсів для користувачів та забезпечення безпеки та інтерактивності на таких платформах.

Метою даної роботи є створення веб-орієнтованої інформаційної системи новинного блогу, що забезпечить ефективне управління контентом та зручний інтерфейс для користувачів.

Для досягнення цієї мети передбачено вирішення наступних задач:

1. Аналіз предметної області та оцінка подібних вже існуючих новинних блогів.

2. Дослідження засобів реалізації основного функціоналу системи.

3. Моделювання етапів роботи веб-додатку, його структури. Розробка архітектури бази даних для збереження інформації, яка буде використана в системі.

4. Практична реалізація веб-додатку та інтеграція всіх компонентів системи для забезпечення безперебійної роботи та зручності використання.

Результати цього дослідження можна використати для вдосконалення функціоналу існуючих веб орієнтованих систем або розробки нових інформаційних систем. Дослідження служить підвищенню ефективності новинних платформ, покращенню взаємодії з користувачами та створенню більш безпечного і зручного середовища для обміну інформацією. Веб додаток, який буду розроблений може стати підґрунтям для подальших інновацій у галузі інформаційних технологій, зокрема зі сторони інформаційної безпеки та покращенню зручності користування веб-сервісами.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень та публікацій

Аналіз предметної області є ключовим етапом у підготовці дипломної роботи, оскільки він визначає загальний напрям дослідження та формує підґрунтя для подальшої роботи. У процесі аналізу відкривається глибше розуміння теми дослідження та виявляються ключові питання, завдання та гіпотези, на яких базується вся робота. Якісно проведений аналіз предметної галузі дає можливість не лише зібрати необхідну інформацію для написання дипломної роботи, а й сформувати повноцінне уявлення про досліджуваний об’єкт, виявити його актуальні аспекти та проблемні зони, а також визначити майбутні перспективи для подальших наукових розробок. Такий аналіз покращує визначення межі досліджень, уточнює мету та завдання роботи, а також допомагає знайти найбільш ефективні методи до вирішення поставлених проблем. [1]

Створення блогу є потужним інструментом для залучення аудиторії, зміцненню позицій у пошукових системах і підвищенню видимості бренду в мережі. Однак важливо врахувати основні етапи розробки веб додатку, щоб блог був не тільки ефективним, а й органічно вписувався в загальну структуру онлайн-ресурсу. Блог дозволяє ділитися експертними знаннями, створювати корисний і цінний контент для цільової аудиторії, а також вибудовувати довірчі відносини з користувачами.

Для того щоб блог став справжнім інструментом успіху необхідно не лише розпочати його ведення, а й ретельно продумати кожен етап цього процесу. Ведення блогу це не лише оприлюднення статті, а й складна та систематична робота, що потребує чіткої стратегії, планування та регулярної взаємодії з аудиторією. Потрібно розробити план, врахувати інтереси та потреби читачів, а також проводити аналіз кожної статті. Справжня цінність

блогу полягає в регулярній актуалізації контенту, отриманню зворотного зв'язку від підписників, а також у створенні платформи для обговорень. Це дає можливість не тільки утримувати існуючу аудиторію, але й залучати нових користувачів, перетворюючи блог на стратегічний елемент розвитку онлайн-платформи. [2]

Найбільше кількість відвідувачів приваблюють новинні блоги, блоги, пов'язані з хобі та бізнес-блоги. Ранні блоги були простими колекціями особистих статей і думок авторів. Однак сучасні блоги значно еволюціонували та перетворилися на повнофункціональні веб системи. Дійсні блоги включають не тільки список статей, а й розділення статей по категоріям, надають можливість залишати коментарі та взаємодіяти з мультимедійними елементами та інтерактивними функціями. З часом блоги набули різноманітних видів та функціональних можливостей. Дехто. Використовує їх як особистий майданчик для самовираження, хтось як інструмент для бізнесу, маркетингу. Сучасний блог може включати не лише текстові матеріали, а й мультимедійні елементи. Це перетворення робить блоги більш привабливими для широкої аудиторії і дає змогу більш ефективно взаємодіяти з користувачами. [3]

Системи новинних блогів є популярними платформами для публікації статей, обміну інформацією та комунікації. Блоги служать як для оповіщення, так і для залучення нових користувачів через інтерактивні елементи, такі як коментарі та рейтинги статей. Новинні блоги часто охоплюють велику аудиторію та тем, зокрема політику, розваги, спорт і технології. Зазвичай наповнюють сайт редактори. Читачам дозволено висловлювати свої думки в коментарях та ділитися статтями в соціальних мережах.

Запуск новинного блогу може стати не лише корисним, а й ефективним кроком. Не беручи до уваги досвід у журналістиці існує кілька важливих причин для створення блогу. Однією з них є можливість об'єднати спільноту зацікавлених осіб та привернути увагу нових читачів. [4]

Порівняльна характеристика блогів на різну тематику наведена у таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика

Характеристика	**Новинний блог**	**Особистий блог**
Цільова аудиторія	Широка аудиторія, майже всі категорії користувачів	Особисті читачі за певними інтересами
Залученість	Висока (коментарі, соціальні мережі)	Середня (вузьке направлення має менше взаємодії)
Тип контенту	Статті, репортажі, інтерв'ю	Особисті роздуми, рецензії, досвід
Частота публікації	Дуже висока	Зазвичай нерегулярна, в залежності від часу та натхнення

Джерело: розроблено автором

Новинний блог — це джерело, що націлене на постійне оновлення інформації з різних сфер (політика, економіка, культура, наука та інші). Основною метою його створення є донесення до аудиторії інформації про новітні події та напрями. Для новинного блогу притаманна висока частота публікацій, неупередженість викладу матеріалу та велика кількість інтерактивних елементів.

В свою чергу, особистий блог є більш невимушеним простором для автора, де він має можливість ділитися своїми думками, роздумами та поглядами на різноманітні теми. Зазвичай такий блог є не сильно популярним, але аудиторія, яка його читає може бути дуже лояльною та активно залученою. Як правило, контент в особистому блозі є більш суб'єктивним, що дає змогу автору встановлювати глибший зв'язок зі своїми читачами. Однією з важливих

рис особистих блогів є їхня неформальність: кількість та частота публікацій може змінюватися в залежності від натхнення автора, а стиль викладення матеріалу може коливатися від глибоких роздумів до легких для прочитання статей. [5]

1.2 Аналіз програмних продуктів-аналогів

Аналіз конкурентів — це метод вивчення інших компаній, які пропонують подібні продукти або послуги. Такий аналіз важливий для того, щоб визначити, як можна вдосконалити власний бізнес і залучити нових користувачів. [6]

Існує широкий вибір рішень для ведення новинних блогів — від простих платформ для створення веб сайтів до складних систем для медіа-порталів. Вони мають різні особливості, такі як формат запуску сайту, наявність технічної підтримки, вбудовані функції та свободу у веденні блогу. Наприклад, за допомогою конструктора сайту можна швидко розробити проект, але це буде означати залежність від платформи. Тому перед розробкою сайту слід чітко провести аналіз платформ та вибрати найбільш оптимальне рішення. Якщо потрібно створити простий блог на популярній платформі, оптимальним вибором буде один із сервісів для ведення блогів. А якщо важлива більша свобода дій, варто звернути увагу на готові системи управління контентом. [7]

Для ефективної роботи веб орієнтованої системи потрібно забезпечити адміністраторам сайту оперативний доступ до всього функціоналу, який розробляється та надати зручний та інтуїтивно зрозумілий інтерфейс. З метою побудови чітких вимог до майбутнього проекту було проведено аналіз програмних продуктів-аналогів, таких як «Blogger» та «LiveJournal».

Першою альтернативою є платформа «Blogger» [8]. Blogger – це зручна платформа, яка дуже проста у використанні, що робить її ідеальним рішенням для початківців. Є можливість одразу монетизувати блог через інтеграцію зі службами Google, такими як AdSense. [9]

Сервіс Blogger є ідеальним інструментом для розробки та адміністрування блогів. Платформа є тісно пов`язаною із багатьма соціальними мережами та сервісами від компанії Google, що надає йому дуже потужних та ефективних особливостей. [10]

Головна сторінка додатку представлена на рисунку 1.1.

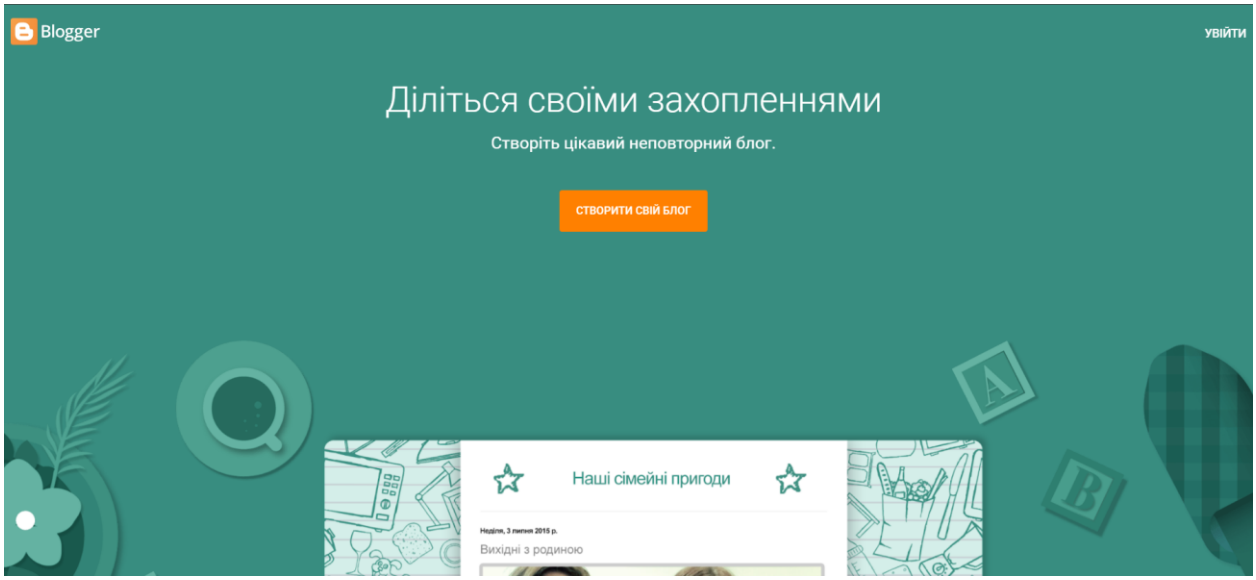

Рисунок 1.1 – Головна сторінка веб-додатку Blogger

Джерело: [8]

Інша альтернатива, яка буде розглянута – це LiveJournal [11]. Це досить популярна і зручна платформа для створення блогу з можливістю коментування. Платформа досить проста, і створити блог на ній може будь-хто, тим паче для цього не потрібно володіти спеціальними навичками.

До очевидної переваги використання платформи можна віднести те, що це є абсолютно безкоштовно. Проте за додаткову плату є можливість підключення додаткових функцій. Платформа є інтуїтивно зрозумілою та зручною у використанні та освоєнні. До недоліків можна віднести дещо

застарілий інтерфейс та зберігання великого обсягу даних лише через перехід на платний тариф. [12]

У результаті проведеного аналізу додатків-аналогів можна представити проміжні підсумки, які зазначені в таблиці 1.2

Таблиця 1.2 – Порівняльна таблиця характеристик додатків-аналогів

	Blogger	**LiveJournal**
Складність налаштування	Низька	Низька
Тип хостингу	Хостинг на сервері	Хостинг на сервері
Гнучкість сайту	Низька	Низька
Безпека даних	Середня (підтримка від Google)	Середня (основні функції безпеки)
Користувацька підтримка	Середня	Помірна
Інтерфейс адміністратора	Простий	Простий
Підтримка шаблонів	Обмежена кількість	Обмежена кількість

Джерело: розроблено автором

Більшість поширених платформ підходять для створення блогів, але кожна з них все одно буде мати у своєму функціоналі зайві інструменти, які неможливо буде вимкнути або прибрати. До суттєвого недоліку це віднести складно, але орієнтуванню в панелі адміністратора ця інформація точно буде зайвою.

З таблиці 1.2 можна зробити висновок, що існуючі додатки-аналоги, які представлені на ринку не відповідають вимогам для виконання поставленого завдання на розробку. У зв’язку з цим було прийнято рішення розробити власну веб-орієнтовану систему, яка дозволить усунути наявні недоліки та

додати необхідний функціонал. Ключовими факторами, що можуть виділити наш блог серед конкурентів, є інтуїтивно зрозумілий інтерфейс, висока якість контенту та активна спільнота. Також можна віднести унікальні функції, такі як динамічна система коментування та переглядів.

Аналіз предметної області показує, що розробка веб орієнтованої системи новинного блогу потребує чіткого підходу та реалізації функціоналу, орієнтованого на запити користувачів. Для користувачів потрібно забезпечити зручний доступ до контенту, можливість для взаємодії з публікаціями та ефективне управління контентом для адміністраторів. Цей проект має потенціал стати успішною платформою для обміну інформацією та думками, задовольняючи потреби користувачів та викликів сучасного ринку. [13]

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Основні завдання

Мета дослідження: Розробка веб орієнтованої інформаційної системи новинного блогу, яка забезпечить користувачам можливість зручного доступу до статей, активної участі в обговореннях та адміністрування контенту. Система повинна відповідати сучасним вимогам щодо зручності, безпеки та функціональності.

Визначено такі задачі:

1. Аналіз вимог користувачів:

- визначити основні потреби та очікування різних типів користувачів (адміністратори, звичайні користувачі);
- збір інформації про вимоги розроблюваної системи.

2. Проектування архітектури системи:

- розробити структурну схему бази даних, що враховує всі необхідні таблиці та зв’язки між ними;
- визначити технології, які будуть використані для розробки.

3. Розробка інтерфейсу користувача:

- створити макети основних сторінок (головна сторінка, сторінка статті, панель адміністратора) з акцентом на зручність та доступність;
- впровадити адаптивний дизайн для комфортного використання на різних пристроях.

4. Функціональні можливості категорії «користувачі»:

- можливість перегляду найпопулярніших та нещодавно доданих статей;
- можливість пошуку статті та переглянути список статей за категоріями;
- можливість прокоментувати кожну наявну статтю.

5. Функціональні можливості категорії «адміністратори»:

- додавання, видалення та редагування статей;
- додавання, видалення та редагування інформації про користувачів;
- додавання, видалення та редагування категорій статей;
- можливість завантаження картинок для статті на сайт.

6. Реалізація функціоналу:

- розробити модулі для авторизації та реєстрації користувачів;
- реалізувати функції створення, редагування та видалення статей, коментарів і категорій;
- забезпечити динамічну зміну кількості переглядів статей та впровадити систему коментування.

7. Тестування системи:

- провести функціональне тестування всіх модулів системи для виявлення помилок і недоліків;
- організувати тестування системи з реальними користувачами для отримання зворотного зв'язку та покращення інтерфейсу.

8. Аналіз результатів:

- оцінити ефективність реалізованої системи на основі тестування всього функціоналу;
- провести аналіз можливостей подальшого розвитку та вдосконалення системи.

Ця структура задач дозволить не лише реалізувати проект у встановлені терміни, але й забезпечить його відповідність потребам кінцевих користувачів, створюючи зручний та функціональний новинний блог.

Веб-орієнтована система буде розроблена з використанням фреймворку Yii2. [14] Для роботи в додатку буде використовуватися реляційна база даних – MySQL. [15]

Карта сайту (або sitemap) – це документ, який включає у собі перелік сторінок. Як зміст у великому тексті допомагає читачеві зорієнтуватися в його послідовності, так і карта сайту сприяє пошуковим системам коректно індексувати сторінки та документи веб ресурсу. [16]

Нижче наведена карта сайту розроблюваної системи.

1. Головна сторінка
   - Заголовок сайту
   - Меню навігації:
     - Головна
     - Список всіх доступних статей
     - Список найпопулярніших статей
     - Список нещодавно доданих статей
     - Список категорій:
       - Політика
       - Спорт
       - Технології
       - Культура та інші
2. Сторінка статті
   - Категорія статті
   - Заголовок статті
   - Текст статті
   - Дата публікації
   - Коментарі
   - Кнопка «Залишити коментар»
3. Сторінка категорії
   - Список статей по категорії
   - Пагінація (сторінки)
4. Адміністративна панель
   - Управління користувачами:
     - Додавання нових користувачів в систему

- Редагування інформації про вже наявних користувачів
- Видалення користувача із системи

- Управління категоріями:
  - Додавання нової категорії на блог
  - Редагування інформації про вже наявну категорію
  - Видалення категорії із блогу

- Управління статтями:
  - Додавання нових статей на блог
  - Редагування інформації, яка зазначена у статті
  - Видалення статті і всієї інформації пов'язаної з нею із системи

Макет головної сторінки додатку представлено на рисунку 2.1.

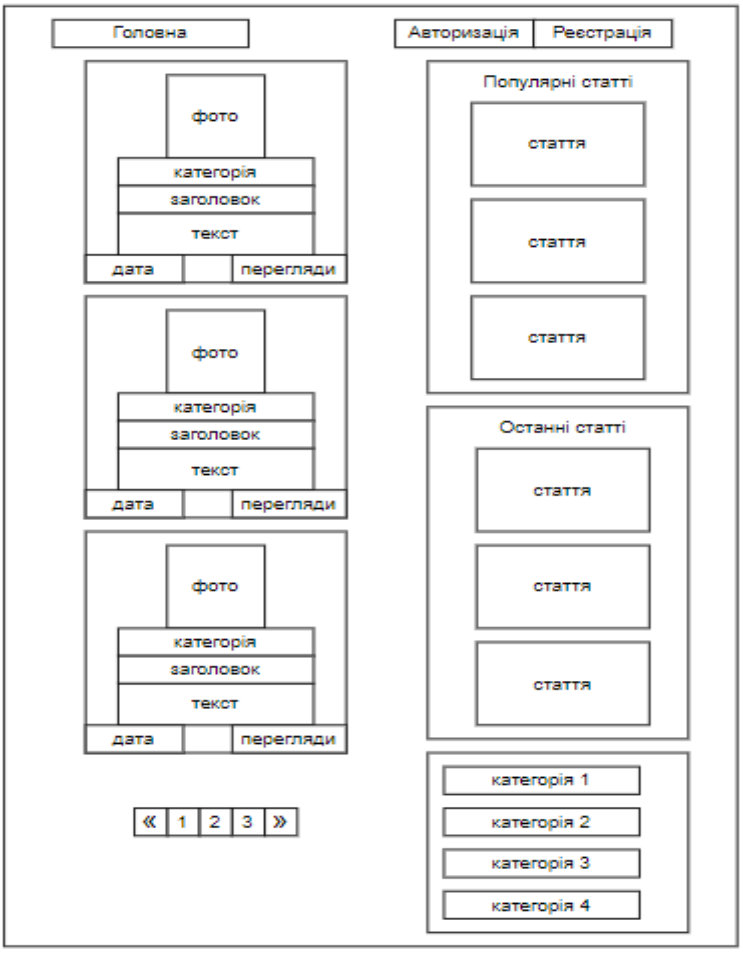

Рисунок 2.1 – Головна сторінка

Джерело: розроблено автором

Макет сторінки статті представлено на рисунку 2.2.

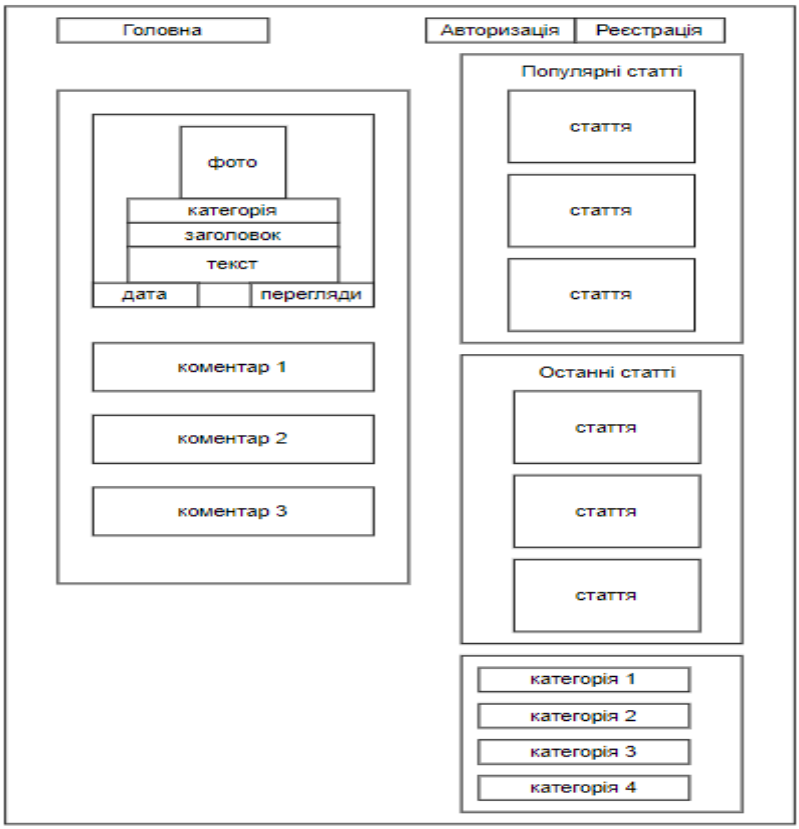

Рисунок 2.2 – Сторінка статті

Джерело: розроблено автором

Макет сторінки категорії представлено на рисунку 2.3.

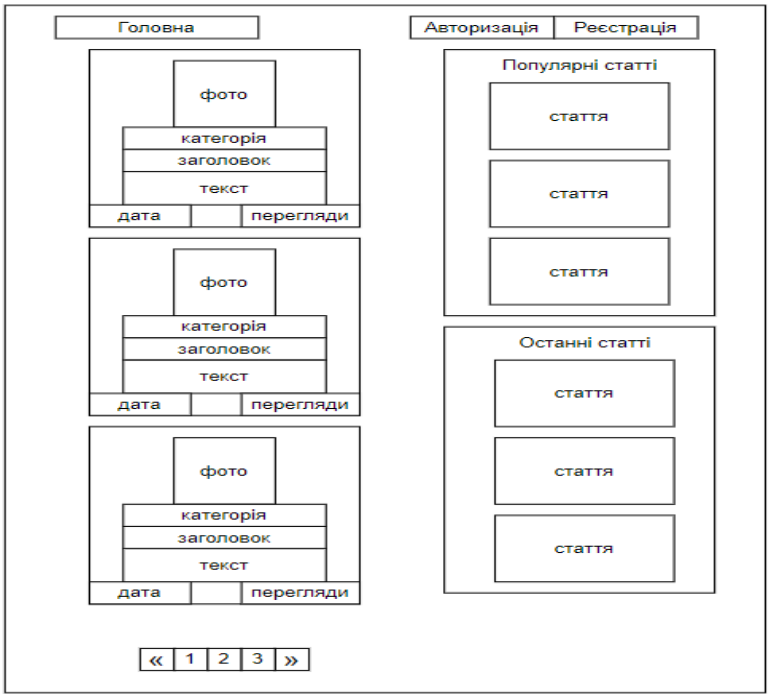

Рисунок 2.3 – Сторінка категорії

Джерело: розроблено автором

Макет панелі адміністратора представлено на рисунку 2.4.

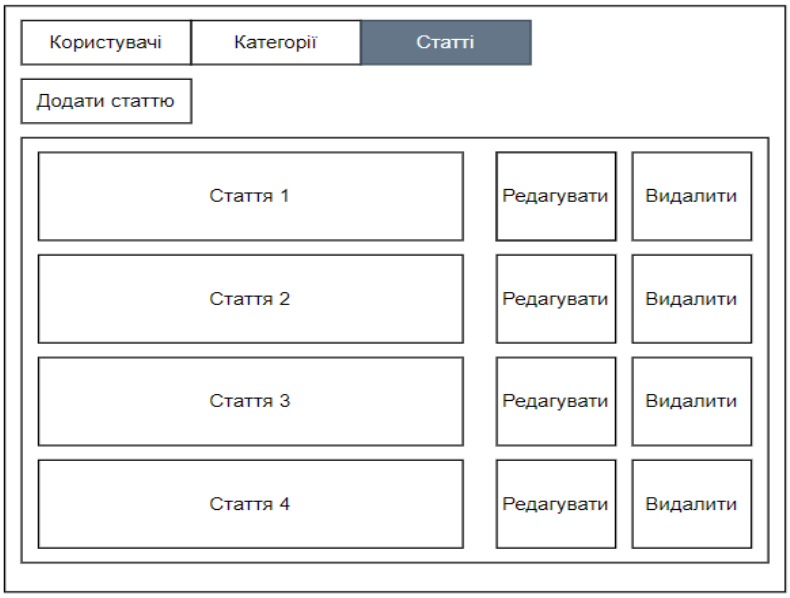

Рисунок 2.4 – Панель адміністратора

Джерело: розроблено автором

2.2 Вибір засобів реалізації основного функціоналу системи

Фреймворк — це визначений набір інструментів та бібліотек, які використовуються для створення програмних додатків.

При розробці сайтів, веб-додатків та інших програмних продуктів є задачі, які потребують рішення, а саме маршрутизація, автентифікація користувачів, кешування, захист від кібератак. Без використання фреймворків майже однаковий код кожен раз потрібно було б писати з початку. Використання фреймворку ж суттєво спрощує цю задачу для розробника та надає безліч переваг:

- Не потрібно створювати кожного разу код з нуля;
- Суттєве прискорення процесу розробки;
- Полегшення роботи з веб додатками;
- Концентрація на логіці, характерній для певної веб-програми;
- Забезпечення базової функціональності веб-продуктів.

Велика частина фреймворків є абсолютно безкоштовними, при цьому їх використання значно покращує продуктивність додатку, оптимізують робочий процес та допомагають набагато швидше отримати бажану систему, яка розробляється. Зазвичай розробники не обмежуються вивченням та використанням лише якогось одного фреймворку. Навпаки, під кожну окрему задачу вони використовують найбільш доцільне рішення. [17]

Для проведення дослідження фреймворків, які можуть бути застосовані для розробки веб орієнтованих систем, було вирішено виконати порівняльний аналіз популярних інструментів.

Кількість технологій, фреймворків та інструментів у індустрії розробки веб додатків стрімко зростає.

Наприклад, JavaScript має різноманітну кількість фреймворків для веб розробки. Однак Angular і React зберігають свої лідерські місця у верхній

частині цього списку, оскільки багато розробників вважають їх найкращими інтерфейсними фреймворками розробки через популярність. [18]

React — це інтерфейсна бібліотека JavaScript , яка надає можливість створювати інтерфейс користувача з багаторазово використовуваних компонентів інтерфейсу користувача. Щоб забезпечити продуктивне рішення React використовує рендеринг на стороні сервера Це дозволяє розробникам створювати бездоганний UX і в той же час складний UI.

За допомогою React JS користувачі та розробники мають велику кількість переваг у розробці інтерфейсу веб додатку. Нижче представлені деякі з основних переваг React JS, якими може скористатися розробник:

- Простий процес налагодження. Код можна використовувати повторно.
- Легкий у освоєнні завдяки простому дизайну.
- Підтримка Android та IOS платформ.
- Підтримка бібліотеки React Native, яка забезпечує ефективну продуктивність.

Angular — це інтерфейсний фреймворк JavaScript із відкритим вихідним кодом, розроблений і керований командою Google Angular. Angular є найпопулярнішим клієнтським фреймворком для розробки швидко масштабованих і високопродуктивних мобільних і веб додатків за допомогою HTML, CSS і TypeScript. Angular використовує техніку MVC, яка розділяє роботу додатку на частини та значно прискорює початковий час завантаження веб сторінки.

До суттєвих переваг Angular можна віднести:

- Фреймворк пропонує чисту розробку коду та впровадження залежностей.
- Angular має безліч бібліотек, які допомагають створювати надійні шаблонні рішення.
- Використання моделі Plain Old JavaScript Objects (POJO) допомагає зробити структуру коду масштабованою та незалежною.

- Фреймворк розширює синтаксис HTML.

React і Angular — популярні фреймворки на основі JS. За допомогою них можна створювати складні інтерактивні веб додатки. Хоча вони мають деякі подібності, проте деякі відмінні характеристики роблять їх принципово різними. [19]

У результаті проведеного аналізу двох популярних фреймворків можна представити проміжні підсумки, які зазначені в таблиці 2.1

Таблиця 2.1 – Порівняльна таблиця двох популярних фреймворків

	Angular	React
Тип	Повноцінний структурний каркас	Бібліотека на основі JavaScript
Призначення	Розробка динамічних веб-програм	Створення інтерактивних компонентів інтерфейсу користувача
Мова	TypeScript	JavaScript(сценарій JSX)
Розробка та підтримка	Google	Facebook
Front-end підхід до розробки	Розширює функціональні можливості HTML, віддає перевагу візуалізації на стороні клієнта	Використовує XML-подібний синтаксис під назвою JSX, невелика перевага для відтворення на стороні сервера
DOM	Справжня	Віртуальна
Прив'язка даних	Двостороння	Одностороння
Ідеальні варіанти використання	Розробка складних корпоративних програм, прогресивні та односторінкові веб-додатки	Сучасні та великі веб додатки з часто змінними даними, гібридні додатки з нативним рендерингом для пристроїв Android та iOS

Джерело: [19]

Порівняння двох популярних фреймворків Angular і React показує, що обидва мають свої сильні сторони та унікальні можливості. Але, відмінність між ними також суттєва. Перший фреймворк – це легкий і гнучкий інструмент, який працює на компонентах, що задає простий старт для швидкого створення інтерфейсу. Цей фреймворк хороший для малих або середніх проєктів і дає можливість додавати потрібні інструменти, якщо в цьому є необхідність.

В свою чергу, Angular більше підходить для масштабного проєкту, що вимагає реалізації безлічі функцій та інструментів. React є найкращим рішенням якщо проєкт за обсягом невеликий. Гнучкий компонентний підхід та простий старт з JSX дають можливість швидко розгорнути інтерфейс і додати потрібні функції. Angular точно вимагає більше часу для ознайомлення, проте React точно підійде для новачків у веб розробці. Якщо є конкретні потреби в роутингу або управлінні станом, а також готовність використовувати сторонні бібліотеки, ідеальним вибором стане React. Натомість, якщо перевага в наявності основних інструментів, вже вбудованих у фреймворк, то це Angular. [20]

До вибору фреймворку для розробки веб орієнтованої системи потрібно віднестися відповідально. Це критично важливий етап, оскільки він визначає не тільки технічні аспекти реалізації проєкту, але й його подальшу підтримку та масштабованість. У випадку, коли йдеться про розробку серверної частини вебсистеми з фокусом на стабільність, безпеку та інтеграцію з різними базами даних, вибір фреймворку Yii2 буде точно кращим рішенням, ніж використання React або Angular.

Yii2 - це PHP-фреймворк, який розроблений для вирішення проблем сучасних веб розробників. Він має широкий функціонал, що спрощує процес розробки та підвищує продуктивність додатку. Однією з ключових переваг

Yii2 є його висока швидкість роботи, яка дозволяє створювати потужні веб додатки.

До переваг фреймворку Yii2 можна віднести наступні пункти:

- Швидкість розробки. Наприклад, Yii2 генерує код, який автоматизує розробку основних частин веб додатку, таких як моделі, контролери та види. Це значно прискорює час створення, необхідний для написання базового коду, і дає можливість розробникам швидко перейти до роботи над функціоналом веб додатку. Окрім того, Yii2 має вбудовану систему міграцій, що значно спрощує роботу з базою даних. За допомогою цієї системи розробники можуть без зусиль створювати та виконувати міграції, що підвищує ефективність розробки.
- Ефективність. Фреймворк має оптимізовану архітектуру, яка дає можливість досягти високої продуктивності навіть при великій кількості запитів. Фреймворк застосовує різноманітні методи кешування та оптимізації запитів до бази даних, що дозволяє скоротити час відгуку сервера і покращити загальну ефективність веб-застосунку. Крім того, Yii2 містить вбудовану систему контролю версій, яка дає змогу розробникам зручно відслідковувати і вносити зміни в код. Це гарантує стабільну роботу веб-застосунку, навіть якщо над ним працює велика кількість розробників

Yii2 – це потужний інструмент, який дозволяє розробникам створювати високоякісні веб-додатки в стислі строки та забезпечувати їх стабільну роботу навіть за умов високого трафіку. Завдяки своєму багатому функціоналу та оптимізованій архітектурі, Yii2 є відмінним вибором для реалізації будь-якого веб-проекту.

Використовуючи Yii2, розробники можуть значно заощадити час та ресурси, які зазвичай витрачаються на написання коду, і зосередитися на розробці функціональності додатка. Його багатофункціональність, висока продуктивність та ефективна взаємодія з базами даних роблять Yii2 одним з найбільш популярних фреймворків для створення веб-застосунків. [21]

За результатами цього дослідження можна зробити висновок, що при виборі Angular або React потрібно врахувати вимоги до проекту. Перший фреймворк більше підходить під розробку великих систем, де потрібні вбудовані інструменти для розробки. React є доцільним вибором для проектів малих або середніх масштабах, де важливі швидкість та гнучкість в розробці, можливість інтеграції сторонніх бібліотек. Для створення серверної частини веб додатку, де ключову роль відіграє стабільність та інтеграція з базами даних доцільно використати фреймворк yii2. Цей фреймворк гарантує високу продуктивність, ефективно працює з базами даних та має зручні інструменти для розробки в найкоротші терміни, що робить його ідеальним вибором для великих і масштабованих проектів.

3 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ

3.1 Функціональне моделювання веб-орієнтованої системи в IDEF0

DEF є частиною родини мов моделювання, що охоплюють широкий спектр застосувань, від функціонального моделювання до моделювання даних, об'єктно-орієнтованого аналізу та проектування, а також збору інформації. [22]

IDEF0 — це графічна нотація, що призначена для формалізації та опису бізнес-процесів. Основною метою цієї методології є створення функціональної діаграми, яка детально і точно відображає процеси з необхідною деталізацією для повного моделювання системи.

Контекстна діаграма структурно-функціональної моделі процесу взаємодії користувача з блогом в нотації IDEF0 представлена на рисунку 3.1.

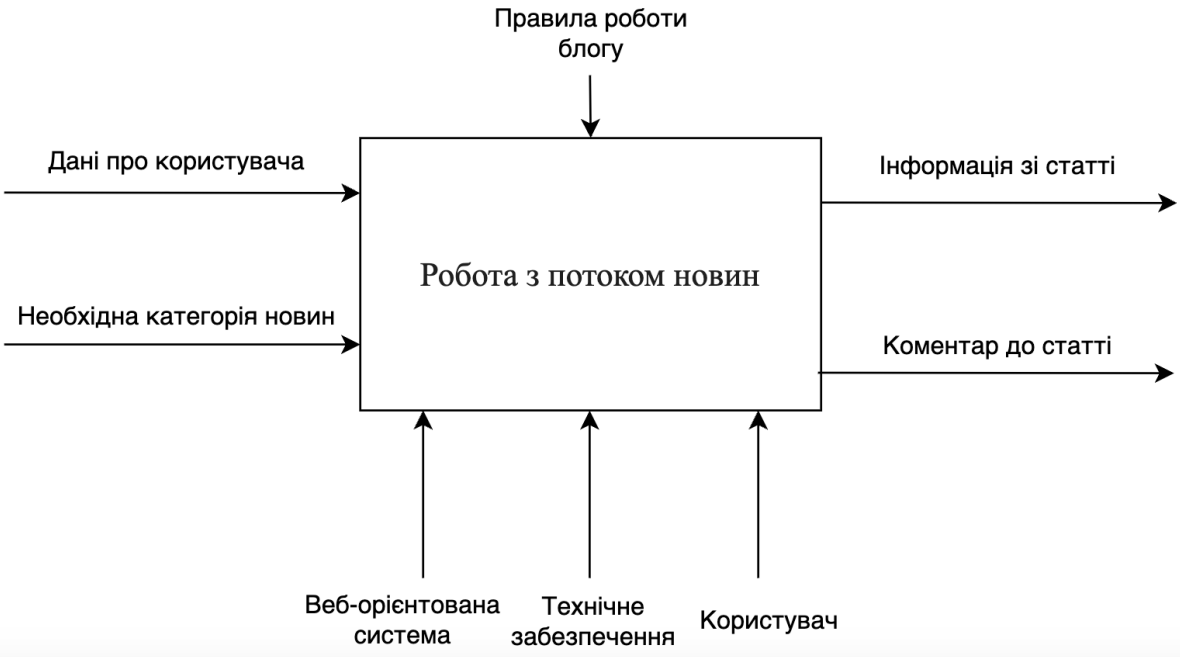

Рисунок 3.1 – Контекстна діаграма IDEF0

Джерело: розроблено автором

На вхід поступають дані про користувача та необхідна категорія новин. Основний процес представлений як робота з потоком новин. Реалізація основного процесу відбуватиметься за рахунок роботи розробленою веб-орієнтованої системи. Обмежують даний процес правила роботи блогу. Окрім цього задіяні ще такі механізми реалізації, як веб-орієнтована система, технічне забезпечення та користувач. На виході маємо інформацію із статті та коментар до статті.

Наступним етапом є декомпозиція контексту, яка передбачає поділ складного процесу на окремі компоненти. Декомпозиція є ключовим елементом стандарту IDEF0. Суть цього підходу полягає в розбитті великого процесу на менші, більш зручні для управління частини. Модель системи в такому випадку представлена в ієрархічній формі діаграм. Схеми IDEF0 зазвичай включають такі компоненти, як контекстна діаграма, батьківська та дочірня схеми, а також дерева вузлів. [23]

Декомпозиція процесу взаємодії користувача з блогом на першому рівні моделювання за допомогою нотації IDEF0 показана на рисунку 3.2.

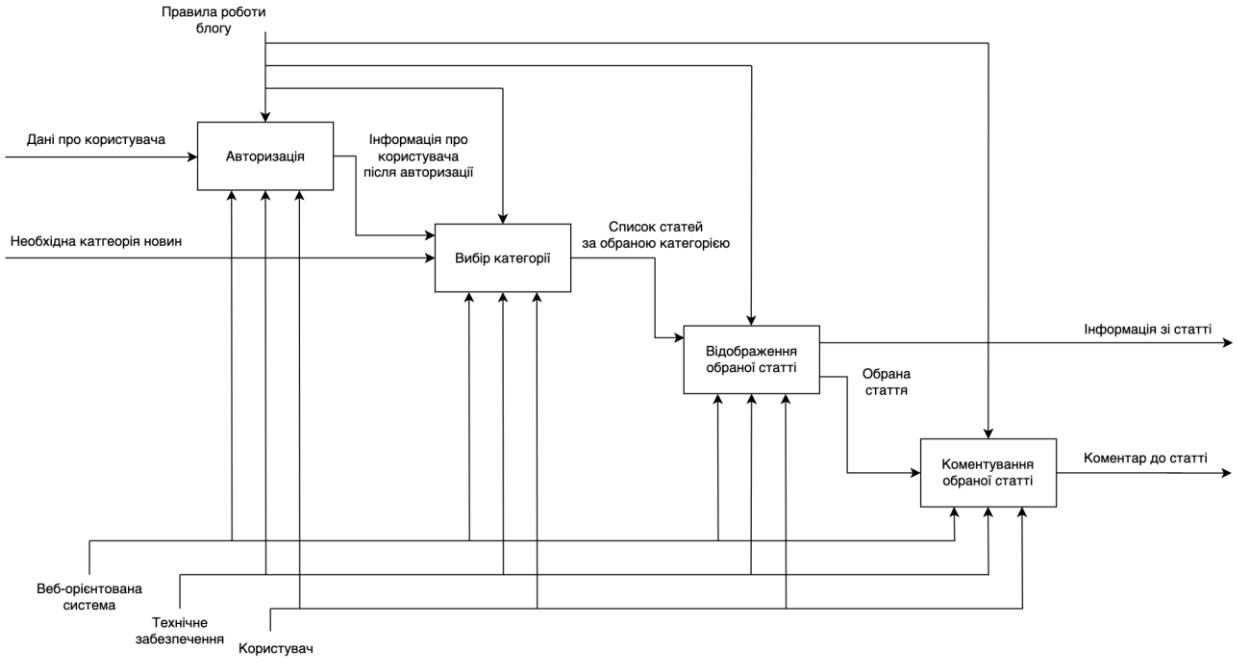

Рисунок 3.2 – Декомпозиція функціональної моделі

Джерело: розроблено автором

Процес взаємодії користувача з блогом складається з таких етапів, як авторизація, вибір категорії, відображення обраної статті та коментування обраної статті. Авторизація – це найперший етап, задача якого полягає у вході в систему унікального користувача. Вибір категорій – це другий етап, який дозволяє користувача обрати статті за категорією. Відображення обраної статті дає можливість повноцінно ознайомитися зі статтею. І останній етап – це коментування обраної статті. Цей етап забезпечує можливість залишити коментар до статті.

3.2 Діаграма варіантів використання

Діаграми варіантів використання допомагають фіксувати вимоги до системи та ілюструють її поведінку в UML. На цих діаграмах описуються високорівневі функції та області застосування системи. Взаємодія між акторами та системою також відображається на цих діаграмах. Зазвичай діаграми варіантів використання розробляються на ранніх етапах проєкту та використовуються як орієнтир протягом усього циклу розробки. [24]

Діаграма варіантів використання в нотації UML представлена на рисунку 3.3.

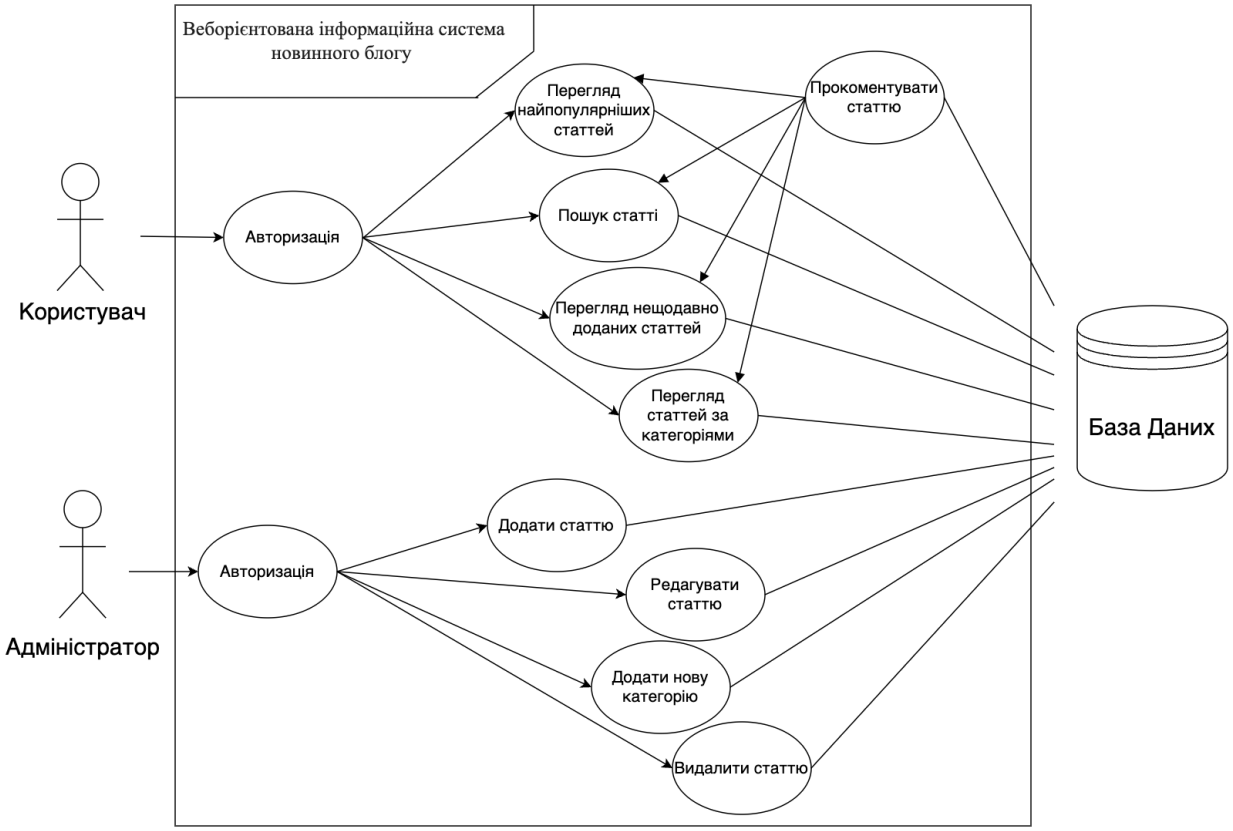

Рисунок 3.3 – Діаграма варіантів використання

Джерело: розроблено автором

У даній системі існує 3 типи акторів:

– Користувач: авторизований користувач, який може коментувати статті, переглядати найпопулярніші та нещодавно додані статті. Також є можливість перегляду статей за категоріями та звичайний пошук статей.

– Адміністратор: має найширший доступ до веб-додатку, а саме може додавати, редагувати, та видаляти статтю. Також має можливість додавати нову категорію на блог.

– База даних: реагує на запити та зміни даних.

3.3 Проектування бази даних

Після створення макетів сторінок веб-орієнтованої системи, необхідно правильно реалізувати базу даних, яка буде використовуватись у процесі роботи. Основною метою фізичного проектування є опис методу фізичної реалізації логічного проєкту. [25]

Для опису фізичної структури бази даних застосовуються моделі даних (рис. 3.4).

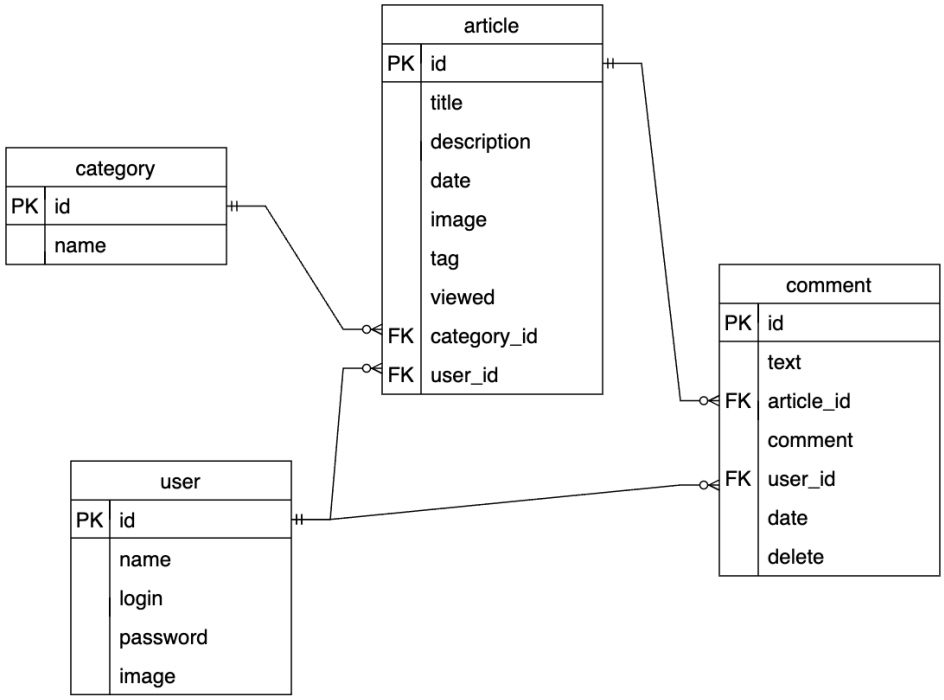

Рисунок 3.4 – Схема бази даних

Джерело: розроблено автором

Таблиця category створена для зберігання даних про категорії статей. Має первинний ключ id та інше поле, а саме назву категорії.

Таблиця user створена для зберігання даних про користувача. Має первинний ключ id та інші поля, а саме ім’я, логін, пароль та фото.

Таблиця article створена для зберігання даних про статтю. Має первинний ключ id та 2 зовнішніх ключа, а саме category_id, user_id. Також є інші поля, а саме заголовок, опис, дата, картинка, тег та кількість переглядів.

Таблиця comment створена для зберігання коментарів. Має первинний ключ id та 2 зовнішніх ключа, а саме article_id та user_id. Також є інші поля, а саме текст, коментар, дата і видалення.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Архітектура додатку

Yii2 застосовує архітектурний шаблон Модель-Представлення-Контролер (MVC, Model-View-Controller), що є популярним у сфері веб-розробки.

MVC сприяє розділенню бізнес-логіки від інтерфейсу, дозволяючи розробнику вносити корективи в окремі компоненти програми без впливу на інші частини. У цій архітектурі модель відповідає за управління даними та логікою, погляд (представлення) — за відображення інтерфейсу користувача (наприклад, текст, поля для введення), а контролер виступає посередником між моделлю та представленням.

Yii2 використовує фронт-контролер, який названий в честь додатку, який визначає контекст оброблення запиту. Веб додаток збирає інформацію про запит і передає її далі. Потім здійснюється подальша обробка за допомогою визначеного контролера. [26]

Структура додатку Yii2 представлена на рисунку 4.1.

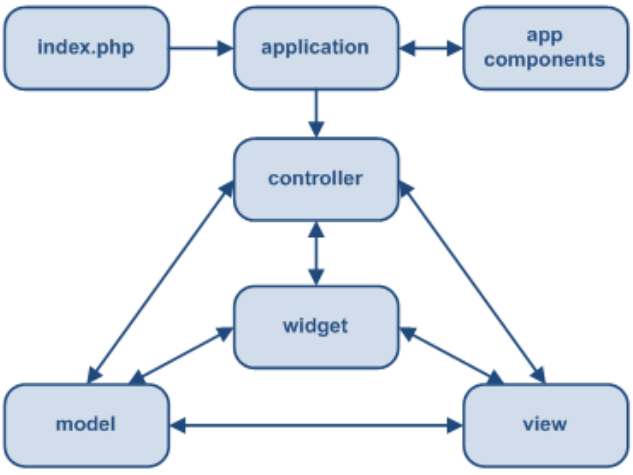

Рисунок 4.1 – Структура додатку Yii2

Джерело: [26]

4.2 Реалізація бази даних

Для роботи з базою даних було обрано MySQL.

Перед створенням бази даних було визначено 4 сутності:

- User
- Category
- Article
- Comment

Для початку роботу з MySQL необхідно запустити PhpMyAdmin на OpenServer та створити там пусту базу даних. Потім, у файлі config/db.php провести налаштування:

```
<?php
return [
    'class' => 'yii\db\Connection',
    'dsn' => 'mysql:host=localhost;dbname=blog',
    'username' => 'root',
    'password' => '',
    'charset' => 'utf8',
];
```

Yii2 має функціонал міграції, тому саме це було і застосовано. Щоб створити міграції, у командному рядку необхідно виконати команду [27]:

```
yii migrate/create create_НАЗВА_ТАБЛИЦІ_table
```

Код міграції для сутності Category:

```
class m231222_084854_create_category_table extends Migration
{
    /**
     * {@inheritdoc}
     */
    public function safeUp()
    {
        $this->createTable('{{%category}}', [
            'id'=>$this->primaryKey(),
            'name'=>$this->string()
        ]);
    }
```

```
    /**
     * {@inheritdoc}
     */
    public function safeDown()
    {
        $this->dropTable('{{%category}}');
    }
}
```

Тепер залишається лише зробити міграцію. Це робиться за допомогою команди:

```
yii migrate
```

Після виконання даної команди з'являється новостворена база даних. Модель бази даних зображена на рисунку 4.2

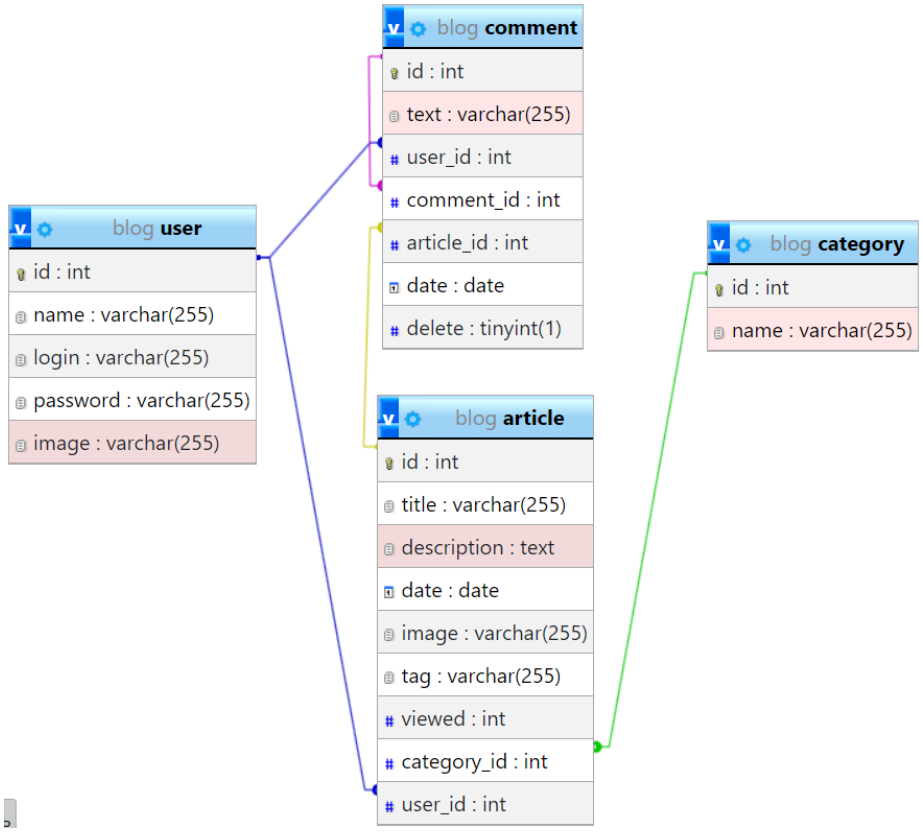

Рисунок 4.2 – Модель бази даних

Джерело: розроблено автором

4.3 Програмна реалізація

Для початку роботу з MySQL необхідно запустити PhpMyAdmin на OpenServer та створити там пусту базу даних. Потім, у файлі config/db.php провести налаштування:

```
'class' => 'yii\db\Connection',
'dsn' => 'mysql:host=localhost;dbname=blog',
'username' => 'root',
'password' => '',
'charset' => 'utf8',
```

Yii2 має функціонал міграції, тому саме це було і застосовано. Щоб створити міграції, у командному рядку необхідно виконати команду [27]:

```
yii migrate/create create_НАЗВА_ТАБЛИЦІ_table
```

Далі будуть наведені коди міграції для кожної сутності.

Код міграції для сутності Category:

```
class m231222_084854_create_category_table extends Migration
{
  public function safeUp()
  {
    $this->createTable('{{%category}}', [
      'id'=>$this->primaryKey(),
      'name'=>$this->string()
    ]);
  }
  /**
   * {@inheritdoc}
   */
  public function safeDown()
  {
    $this->dropTable('{{%category}}');
  }
}
```

Код міграції для сутності User:

```
class m231222_084910_create_user_table extends Migration
{
  public function safeUp()
  {
```

```
        $this->createTable('{{%user}}', [
            'id' => $this->primaryKey(),
            'name' => $this->string(),
            'login' => $this->string(),
            'password' => $this->string(),
            'image' => $this->string()
        ]);
    }
    public function safeDown()
    {
        $this->dropTable('{{%user}}');
    }
}
```

Код міграції для сутності Article:

```
class m231222_084926_create_article_table extends Migration
{
    public function safeUp()
    {
        $this->createTable('{{%article}}', [
            'id' => $this->primaryKey(),
            'title' => $this->string(),
            'description' => $this->text(),
            'date' => $this->date(),
            'image' => $this->string(),
            'tag' => $this->string(),
            'viewed' => $this->integer(),
            'category_id' => $this->integer(),
            'user_id' => $this->integer(),

        ]);
        // create index for column category_id
        $this->createIndex(
            'idx-category_id',
            'article',
            'category_id'
        );
        // add foreign key for table category
        $this->addForeignKey(
            'fk-category_id',
            'article',
            'category_id',
            'category',
```

```
            'id',
            'CASCADE'
        );
        // create index for column user_id
        $this->createIndex(
            'idx-post-user_id',
            'article',
            'user_id'
        );
        // add foreign key for table user
        $this->addForeignKey(
            'fk-post-user_id',
            'article',
            'user_id',
            'user',
            'id',
            'CASCADE'
        );
    }
    public function safeDown()
    {
        $this->dropTable('{{%article}}');
    }
}
```

Код міграції для сутності Comment:

```
class m231222_084943_create_comment_table extends Migration
{
    public function safeUp()
    {
        $this->createTable('{{%comment}}', [
            'id' => $this->primaryKey(),
            'text' => $this->string(),
            'user_id' => $this->integer(),
            'comment_id' => $this->integer(),
            'article_id' => $this->integer(),
            'date' => $this->date(),
            'delete' => $this->boolean(),
        ]);
// create index for column user_id
        $this->createIndex(
            'idx-post-user_id',
            'comment',
```

```
            'user_id'
        );
// add foreign key for table user
        $this->addForeignKey(
            'fk-comment-post-user_id',
            'comment',
            'user_id',
            'user',
            'id',
            'CASCADE'
        );
// create index for column article_id
        $this->createIndex(
            'idx-article_id',
            'comment',
            'article_id'
        );
// add foreign key for table article
        $this->addForeignKey(
            'fk-article_id',
            'comment',
            'article_id',
            'article',
            'id',
            'CASCADE'
        );
// create index for column comment_id
        $this->createIndex(
            'idx-comment_id',
            'comment',
            'comment_id'
        );
// add foreign key
        $this->addForeignKey(
            'fk-comment_id',
            'comment',
            'comment_id',
            'comment',
            'id',
            'CASCADE'
        );
    }
```

```
    public function safeDown()
    {
        $this->dropTable('{{%comment}}');
    }
}
```

Тепер залишається лише зробити міграцію. Це робиться за допомогою команди:

```
yii migrate
```

Після виконання даної команди з’являється новостворена база даних.

Робота з моделями буде відбуватися через Gii – інструмент генерування коду в Yii2 [29].

Спочатку налаштовується підключення до Gii через файл config/web.php:

```
        if (YII_ENV_DEV) {
    $config['bootstrap'][] = 'gii';
    $config['modules']['gii'] = [
        'class' => 'yii\gii\Module',
        // uncomment the following to add your IP if you are not connecting from localhost.
        //'allowedIPs' => ['127.0.0.1', '::1'],
    ];
}
```

Щоб генерувати код, необхідно перейти за посиланням <домен>/gii.

Першим етапом стане створення модуля admin за допомогою Module Generator.

Потім, за допомогою Model Generator створюємо усі моделі (у полі «Table Name» необхідно прописати *), які є в базі даних. Ця операція зображена на рисунку 4.3.

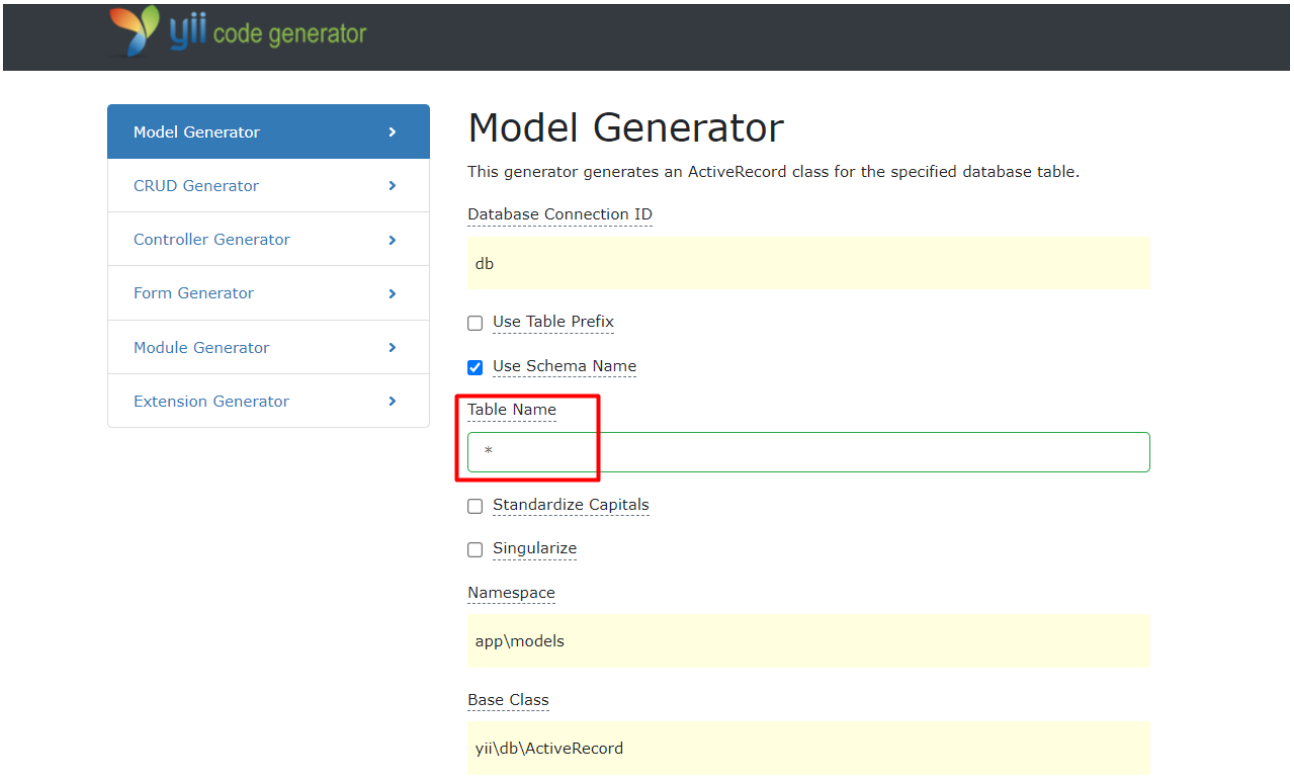

Рисунок 4.3 – Створення моделей за допомогою gii

Джерело: розроблено автором

Далі треба обрати, які із запропонованих сутностей треба створити. Обрати треба усі, окрім Migrate.

Також, gii має інструмент CRUD Generator, який дозволяє створити усі контролер з усіма основними методами CRUD:

- Create
- Read
- Update
- Delete

Розглянемо створення контролера на прикладі сутності Article. Дана операція зображена на рисунку 4.4.

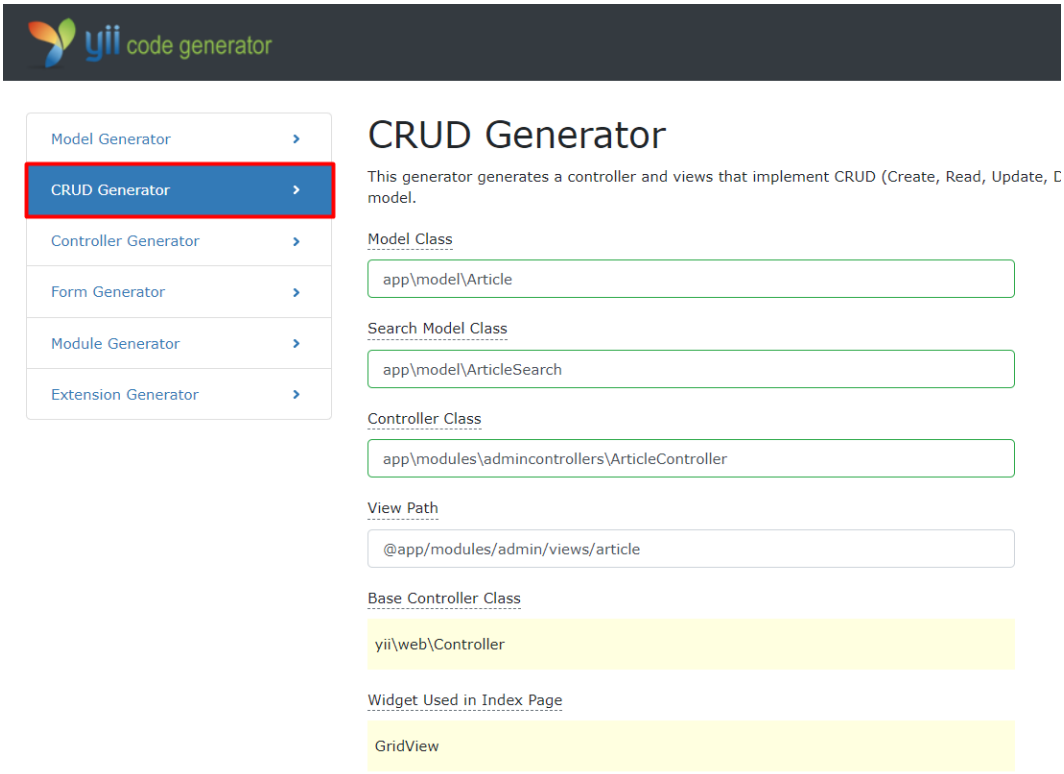

Рисунок 4.4 – CRUD генератор для сутності Article

Джерело: розроблено автором

Таку операцію, як на треба виконати для усіх сутностей проекту.

Після цього в URL прописуємо <домен>/admin/article. Відкриється сторінка на якій можна буде створювати різні статті, редагувати їх та видаляти.

Для того, щоб можна було завантажити картинку для статті, необхідно виконати ряд дій. Далі будуть описані основні.

Спочатку треба додати нову кнопку в modules/admin/views/article/view.php в місце, де знаходяться кнопки оновити та видалити.

```
<?= Html::a('Set Image', ['set-image', 'id' => $model->id], ['class' => 'btn btn-default']) ?>
```

Далі треба створити події в ArticleController:

```
public function actionSetImage($id)
{
  $model = new ImageUpload;
  if (Yii::$app->request->isPost)
  {
    $article = $this->findModel($id);
```

```
        $file = UploadedFile::getInstance($model, 'image');
        if($article->saveImage($model->uploadFile($file, $article->image)))
        {
            return $this->redirect(['view', 'id'=>$article->id]);
        }
    }
    return $this->render('image', ['model'=>$model]);
}
```

У файлі models/Article додаємо методи роботи з фотографіями:

```
        public function getImage()
{
    return ($this->image) ? '/uploads/' . $this->image : '/no-image.jpg';
}
        public function saveImage($filename)
{
    $this->image = $filename;
    return $this->save(false);
}
public function deleteImage()
{
    $imageUploadModel = new ImageUpload();
    $imageUploadModel->deleteCurrentImage($this->image);
}
```

Також, треба створити форму image.php у папці modules/admin/views/article:

```
<div class="article-form">
    <?php $form = ActiveForm::begin(); ?>
    <?= $form->field($model, 'image')->fileInput(['maxlength' => true]) ?>
    <div class="form-group">
        <?= Html::submitButton('Submit', ['class' => 'btn btn-success']) ?>
    </div>
    <?php ActiveForm::end(); ?>
</div>
```

У папці web створюємо папку uploads, куди будуть завантажуватися фото. Вміст папки uploads представлений на рисунку 4.5

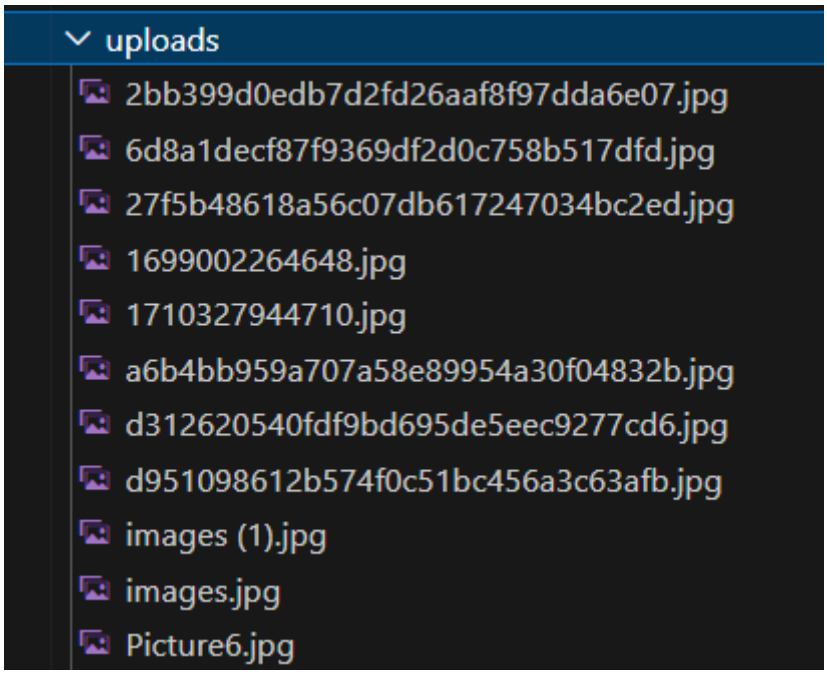

Рисунок 4.5 – Папка з завантаженими фото

Джерело: розроблено автором

Сторінка завантаження фотографії буде мати вигляд як на рисунку 4.6

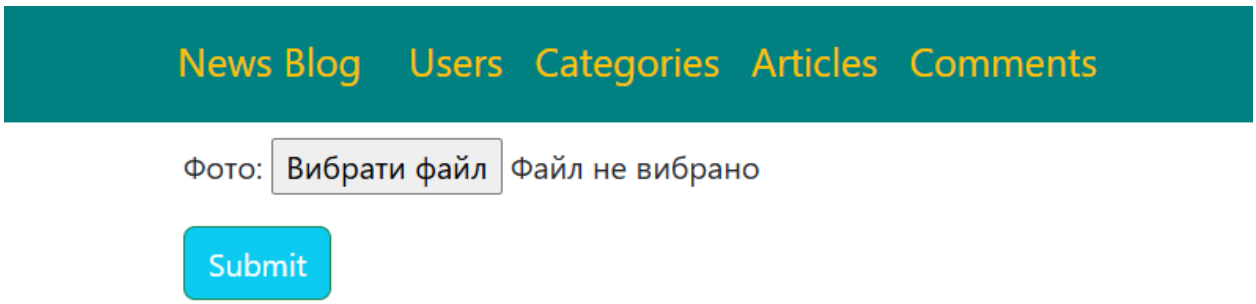

Рисунок 4.6 – Сторінка завантаження фото

Джерело: розроблено автором

Після завантаження фото, у базі даних у таблиці article стовпець image буде мати вигляд як на рисунку 4.7

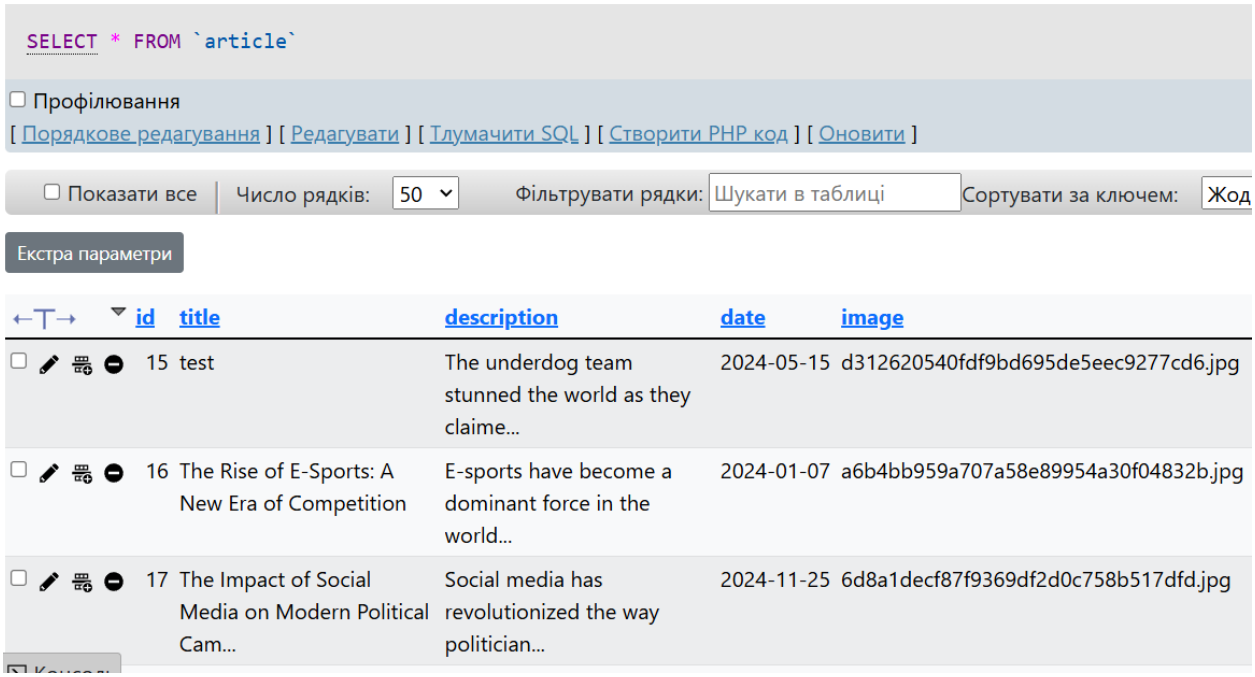

Рисунок 4.7 – Зберігання фото в таблиці article бази даних blog

Джерело: розроблено автором

Розглянемо детально пагінацію. Для того, щоб додати пагінацію, першим кроком є додавання у controllers/SiteController.php у метод actionIndex() наступного коду [29]:

```
            // build a DB query to get all articles
        $query = Article::find();
// get the total number of articles (but do not fetch the article data yet)
        $count = $query->count();
// create a pagination object with the total count
        $pagination = new Pagination(['totalCount' => $count, 'pageSize'=> 1]);
// limit the query using the pagination and retrieve the articles
        $articles = $query->offset($pagination->offset)
            ->limit($pagination->limit)
            ->all();
        return $this->render('index',[
            'articles'=>$articles,
            'pagination'=>$pagination
        ]);
```

У файл views/site/index.php для коректної роботи пагінації треба прописати наступне:

```
<?php echo LinkPager::widget(['pagination' => $pagination,]);?>
```

Пагінація зображена на рисунку 4.8

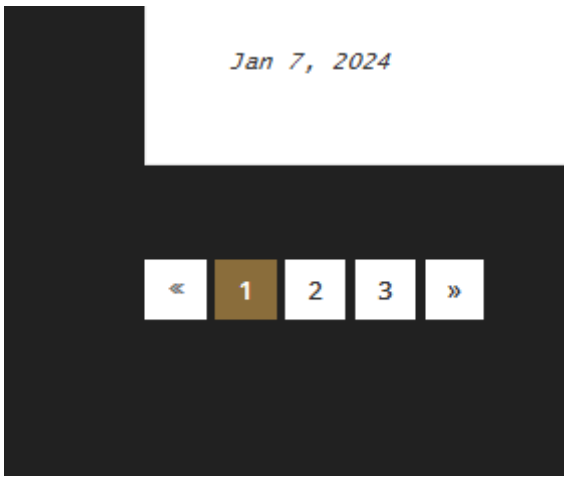

Рисунок 4.8 – Пагінація

Джерело: розроблено автором

Код, який слугує для відображення статтей в адмін панелі представлений в файлі modules\admin\views\article_form.php

```
<div class="article-form">
    <?php $form = ActiveForm::begin(); ?>
    <?= $form->field($model, 'title')->textInput(['maxlength' => true]) ?>
    <?= $form->field($model, 'description')->textarea(['rows' => 6]) ?>
    <?= $form->field($model, 'date')->textInput() ?>
    <?= $form->field($model, 'image')->textInput(['maxlength' => true]) ?>
    <?= $form->field($model, 'tag')->textInput(['maxlength' => true]) ?>
    <?= $form->field($model, 'viewed')->textInput() ?>
    <?= $form->field($model, 'category_id')-
>dropDownList(\yii\helpers\ArrayHelper::map(Category::find()->all(), 'id', 'name')) ?>
    <?= $form->field($model, 'user_id')->dropDownList(\yii\helpers\ArrayHelper::map(User::find()->all(),
'id', 'name')) ?>
    <div class="form-group">
        <?= Html::submitButton('Save', ['class' => 'btn btn-success']) ?>
    </div>
    <?php ActiveForm::end(); ?>
</div>
```

Код, який слугує для створення статтей в адмін панелі представлений в файлі modules\admin\views\article\create.php

```
<div class="article-create">
    <h1><?= Html::encode($this->title) ?></h1>
```

```
    <?= $this->render('_form', [
        'model' => $model,
        'categories' => \yii\helpers\ArrayHelper::map(Category::find()->all(), 'id', 'name'),
        'users' => \yii\helpers\ArrayHelper::map(User::find()->all(), 'id', 'name'),
    ]) ?>
</div>
```

Код, який слугує для оновлення даних в статті представлений в файлі modules\admin\views\article\update.php

```
<div class="article-update">
    <h1><?= Html::encode($this->title) ?></h1>
    <?= $this->render('_form', [
        'model' => $model,
    ]) ?>
</div>
```

Щоб успішно відображалися перегляди на постах спочатку треба додати метод viewedCounter в models/Article.php. Приклад коду представлений нижче.

```
public function viewedCounter()
    {
        $this->viewed += 1;
        return $this->save(false);
    }
```

Наступним кроком є додавання коду представленого нижче до вже існуючого методу actionView, який знаходиться controllers/SiteController.php.

```
$article->viewedCounter();
```

Створюємо файл CommentForm.php в папці /models

```
    public function saveComment($article_id)
    {
        $comment = new Comment;
        $comment->text = $this->comment;
        $comment->user_id = Yii::$app->user->id;
        $comment->article_id = $article_id;
        $comment->date = date('Y-m-d');
        return $comment->save();
    }
}
```

В створений метод actionView, який знаходиться в controllers/SiteController.php додаємо наступний код:

```
$comments = $article->comments;
$commentsParent = array_filter($comments, function ($k) {
```

```
    return $k['comment_id'] == null;
});
$commentsChild = array_filter($comments, function ($k) {
    return ($k['comment_id'] != null && !$k['delete']);
});
$commentForm = new CommentForm();
```

Створюємо метод getDate і додаємо його у вже існуючі файли models/User.php та models/Comment.php

```
public function getDate()
{
    return Yii::$app->formatter->asDate($this->date);
}
```

Змінюємо частину коду, який знаходиться в папці views\site\singleArticle.php

```
<div class="leave-comment"><!--leave comment-->
                <h4>Leave a reply</h4>
                <?php $form = \yii\widgets\ActiveForm::begin([
                    'action' => ['site/comment', 'id' => $article->id],
                    'options' => ['class' => 'form-horizontal contact-form', 'role' => 'form']
                ]) ?>
                <div class="form-group">
                    <div class="col-md-12">
                        <?= $form->field($commentForm, 'comment')->textarea(['class' => 'form-control',
'placeholder' => 'Write Message'])->label(false) ?>
                    </div>
                </div>
                <button type="submit" class="btn send-btn">Post Comment</a>
                    <?php \yii\widgets\ActiveForm::end(); ?>
            </div><!--end leave comment-->
```

4.4 Використання веб-орієнтованої системи

В систему потрібно авторизуватися, щоб використовувати функціонал на повну. Вікно авторизації із заповненими даними зображено на рисунку 4.9

Рисунок 4.9 – Вікно авторизації

Джерело: розроблено автором

Також є можливість зареєструватися в системі. Для цього потрібно заповнити 3 поля: ім'я, логін та пароль. Вікно реєстрації зображено на рисунку 4.10

Рисунок 4.10 – Вікно реєстрації

Джерело: розроблено автором

Авторизувавши в систему в якості адміністратора нам відкривається доступ до додавання, редагування та видалення контенту з сайту. Також є можливість додати картинку до кожної статті. Відкриваємо статтю та обираємо кнопку «Set Image». Вікно обраної статті зображено на рисунку 4.11

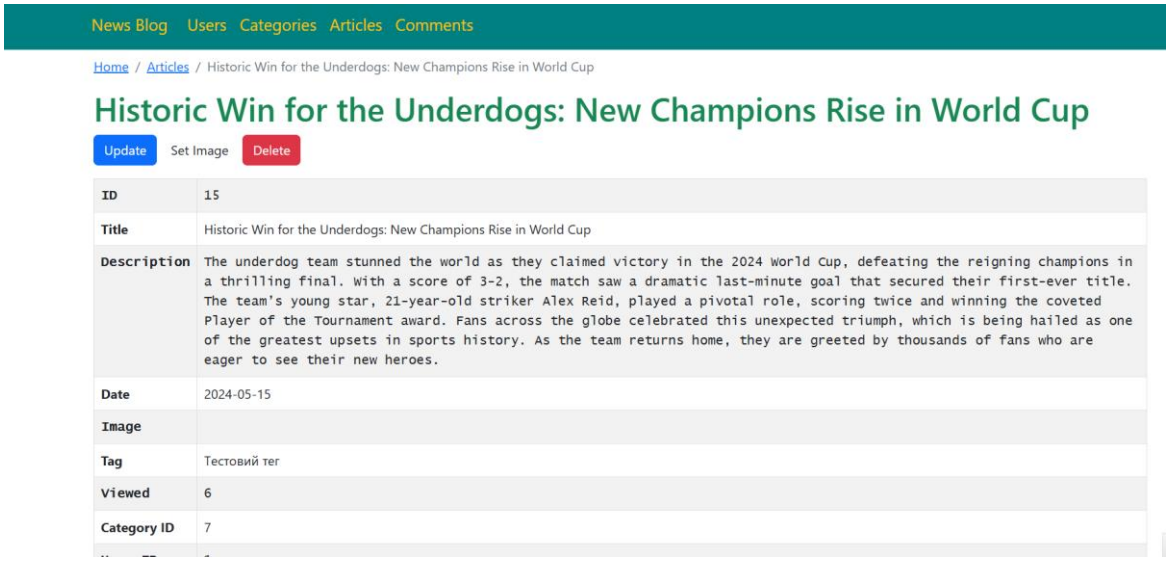

Рисунок 4.11 – Вікно обраної статті

Джерело: розроблено автором

Далі натискаємо «вибрати файл». Обираємо фото, яке потрібно і зберігаємо. Вікно вибору і підтвердження фото зображено на рисунку 4.6

Після успішного додавання фото поле Image змінилось і стало не порожнє. Успішне додавання фото до статті зображено на рисунку 4.12

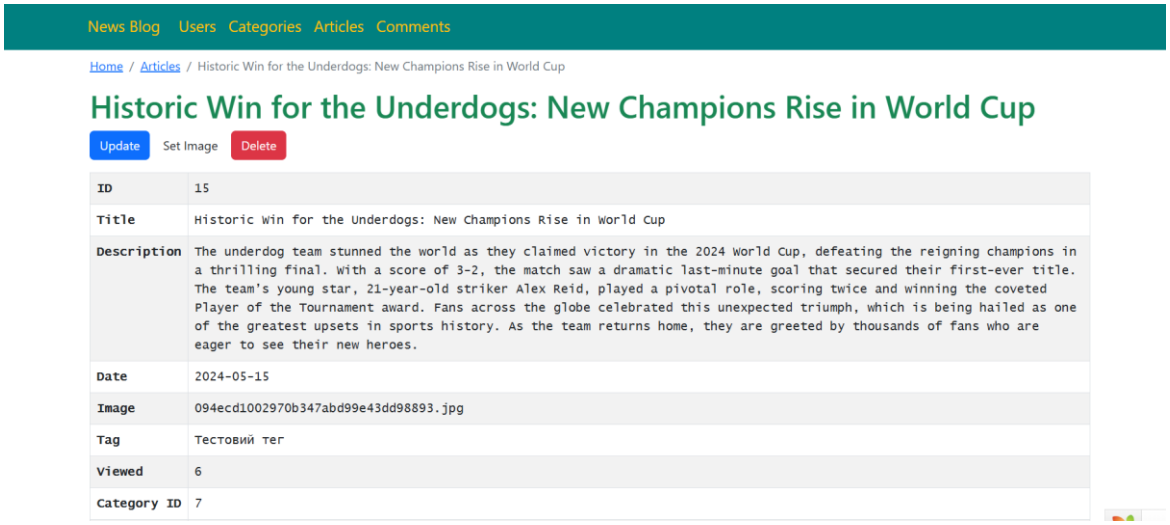

Рисунок 4.12 – Успішне додавання фото до статті

Джерело: розроблено автором

Те що фото додалось коректно можна помітити на головній сторінці, де знаходяться всі статті. Успішне додавання фото до статті на головній сторінці зображено на рисунку 4.13

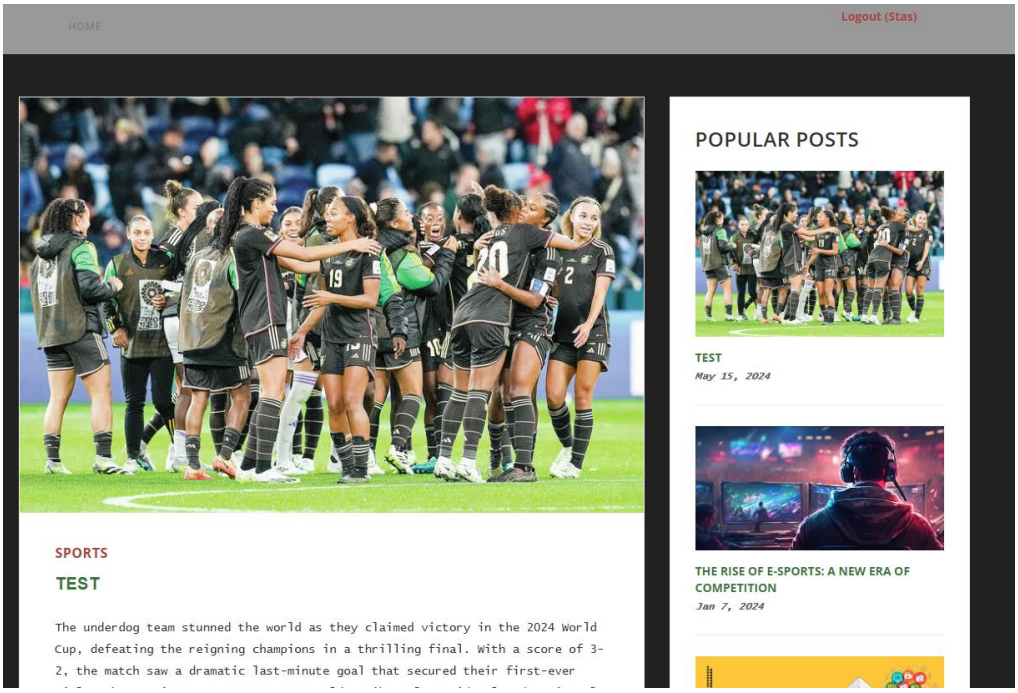

Рисунок 4.13 – Успішне додавання фото до статті на головній сторінці

Джерело: розроблено автором

В панелі адміністратора відкриваємо список користувачів в системі. Список користувачів в системі зображено на рисунку 4.14

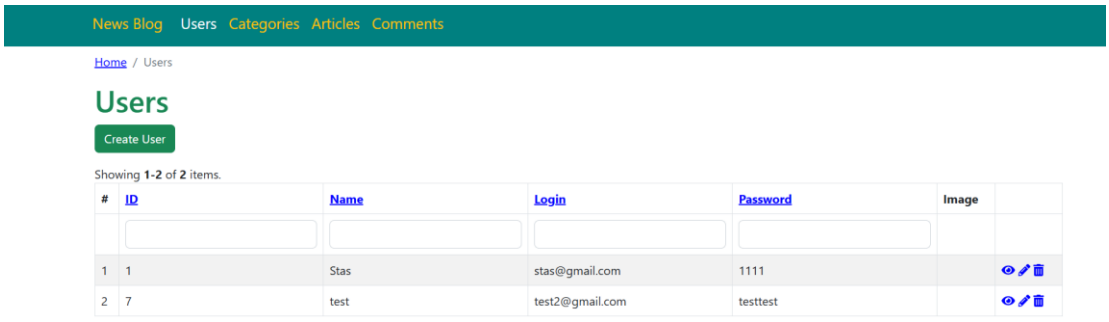

Рисунок 4.14 – Список користувачів в системі

Джерело: розроблено автором

Спробуємо додати нового користувача в систему. Для цього вводимо інформацію у відповідні поля. Натискаємо зберегти. Процес додавання нового користувача зображено на рисунку 4.15

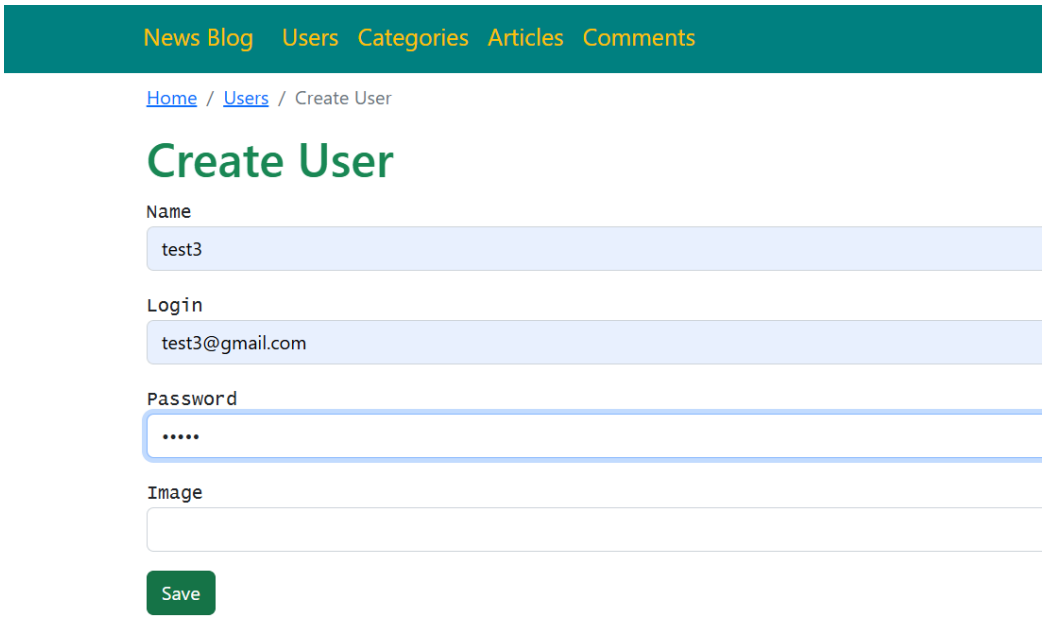

Рисунок 4.15 – Процес додавання нового користувача в систему

Джерело: розроблено автором

Після успішного додавання нового користувача в систему є можливість одразу змінити про нього інформацію. Натискаємо кнопку Update. Процес успішного додавання нового користувача в систему зображено на рисунку 4.16

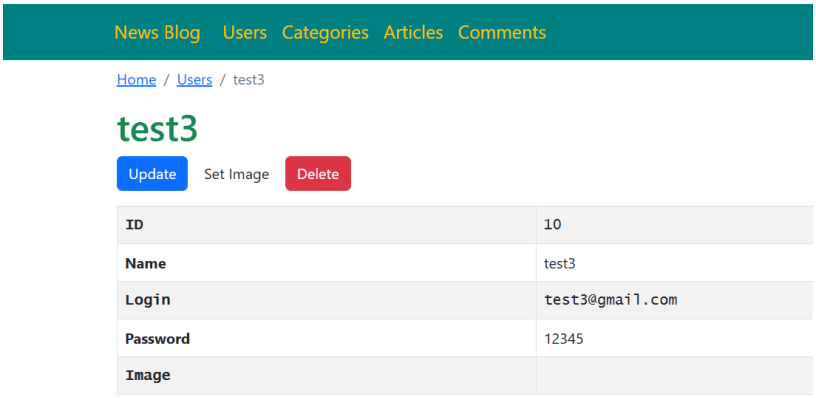

Рисунок 4.16 – Результат успішного додавання користувача в систему

Джерело: розроблено автором

Змінюємо поле Name і натискаємо зберегти. Процес зміни ім’я користувача зображено на рисунку 4.17

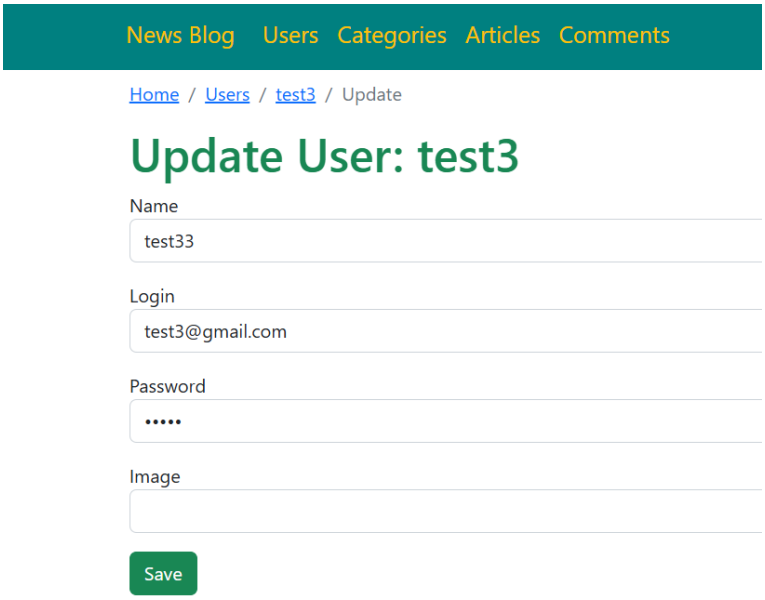

Рисунок 4.17 – Процес зміни ім’я користувача

Джерело: розроблено автором

Результат зміни ім’я користувача зображено на рисунку 4.18

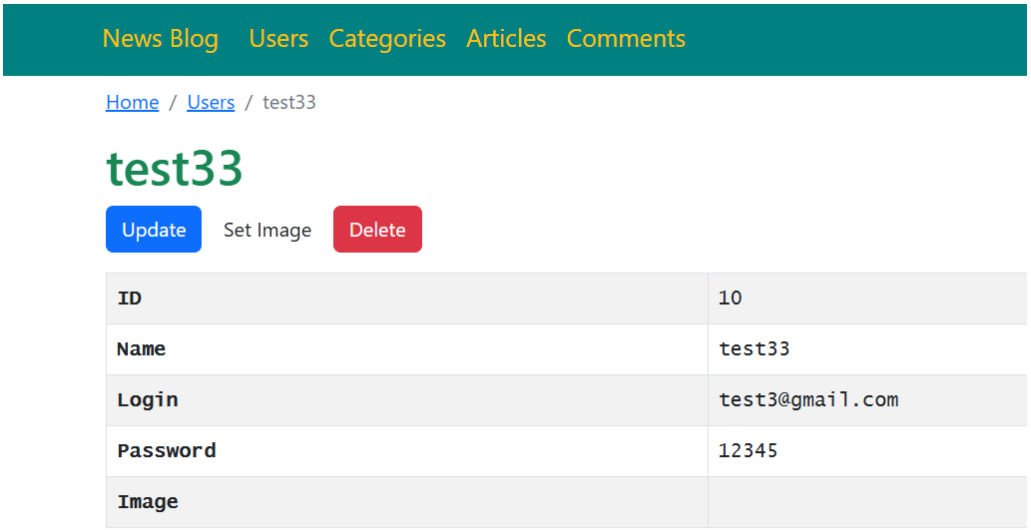

Рисунок 4.18 – Результат зміни ім’я користувача

Джерело: розроблено автором

Відповідний запис з’явився серед загального списку користувачів. Результат зміни серед списку користувачів зображено на рисунку 4.19

Рисунок 4.19 – Результат зміни ім’я серед списку всіх користувачів

Джерело: розроблено автором

Натискаючи кнопку видалити цього користувача з системи вона попереджує про це і повторно запитує. Процес видалення користувача з системи зображено на рисунку 4.20

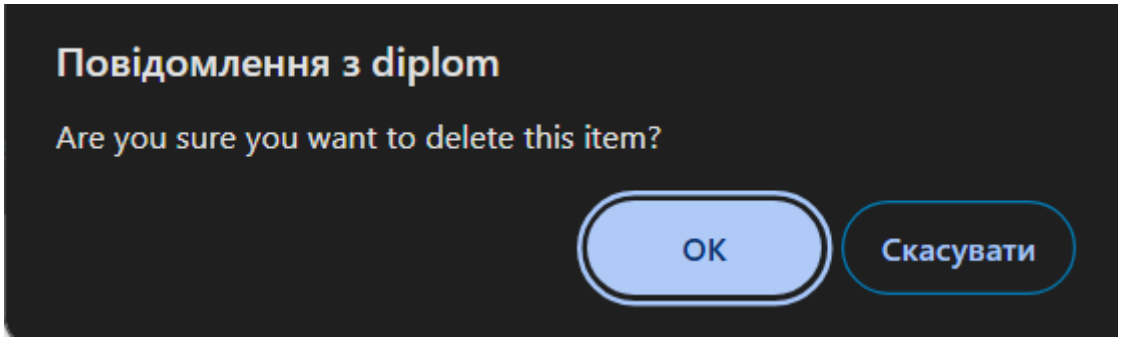

Рисунок 4.20 – Процес видалення користувача з системи

Джерело: розроблено автором

Результат видалення користувача з системи зображено на рисунку 4.21

Рисунок 4.21 – Результат видалення користувача з системи

Джерело: розроблено автором

Наступним етапом є зміна назви статті. Для цього обираємо статтю і натискаємо кнопку «Редагування». Стаття до редагування зображена на рисунку 4.22

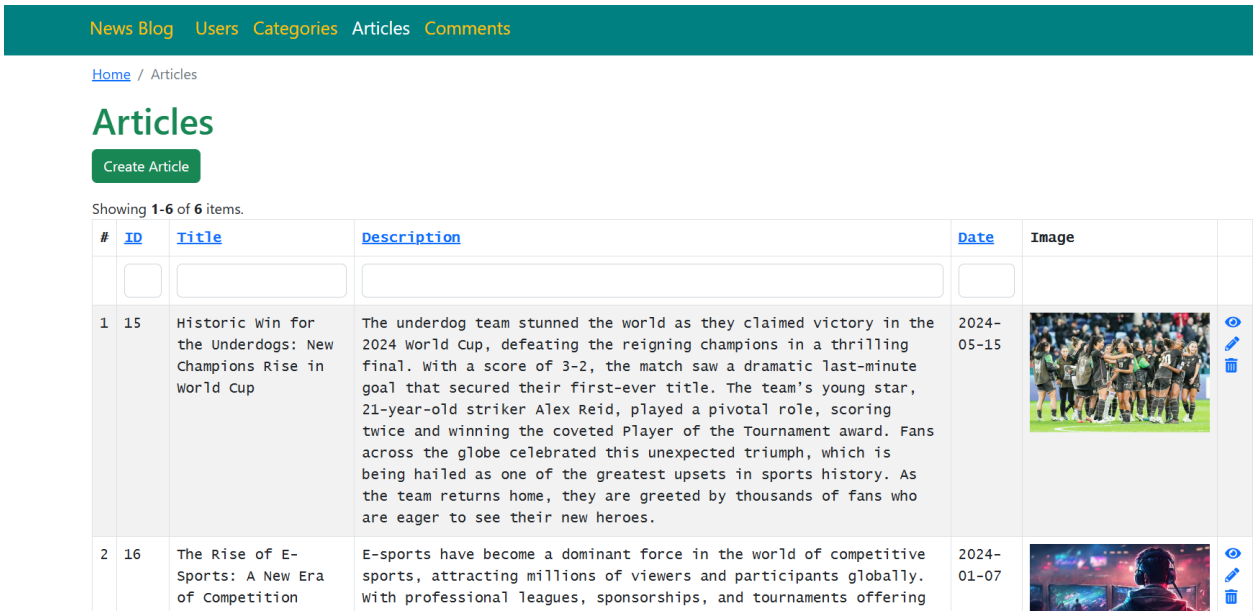

Рисунок 4.22 – Стаття до редагування

Джерело: розроблено автором

Змінюємо назву статті і натискаємо зберегти. Процес зміни назви статті зображено на рисунку 4.23

Рисунок 4.23 – Процес зміни назви статті

Джерело: розроблено автором

Результат зміни назви статті зображено на рисунку 4.24

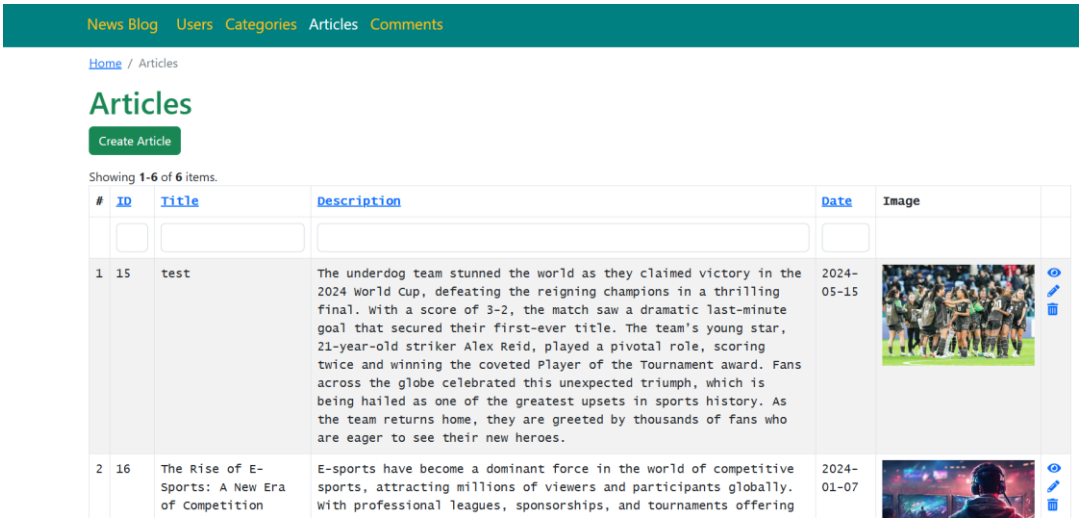

Рисунок 4.24 – Результат зміни назви статті

Джерело: розроблено автором

Розглянемо динамічну зміну кількості переглядів статті. До натискання на статтю маємо 0 переглядів. Стаття до ознайомлення зображена на рисунку 4.25

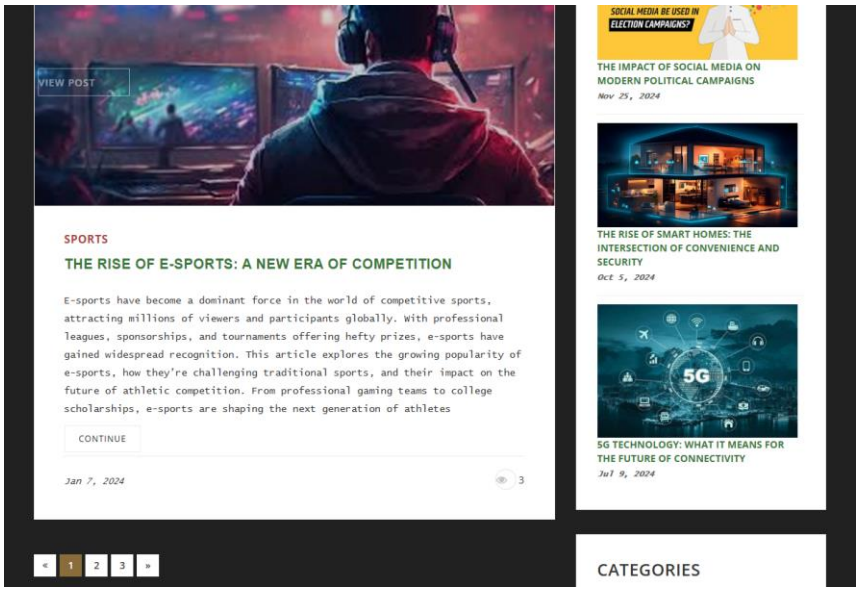

Рисунок 4.25 – Стаття до ознайомлення

Джерело: розроблено автором

Після натискання на статтю одразу змінюється кількість переглядів. Кількість переглядів статті після ознайомлення зображена на рисунку 4.26

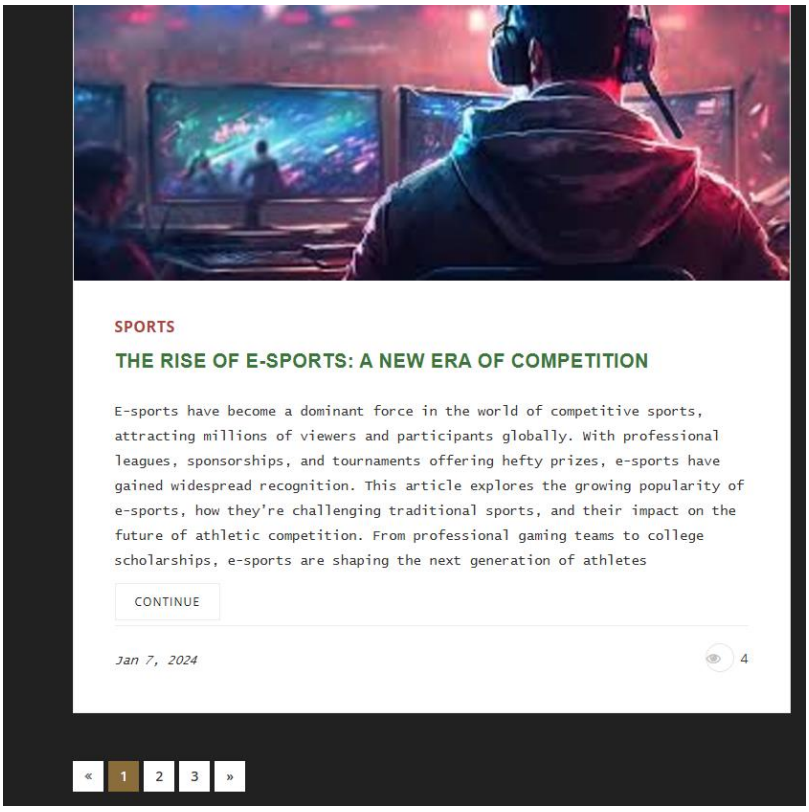

Рисунок 4.26 – Стаття після ознайомлення

Джерело: розроблено автором

Одразу після ознайомлення із статтею вона піднімається в топі найпопулярніших. Список популярних статей на блозі зображено на рисунку 4.27

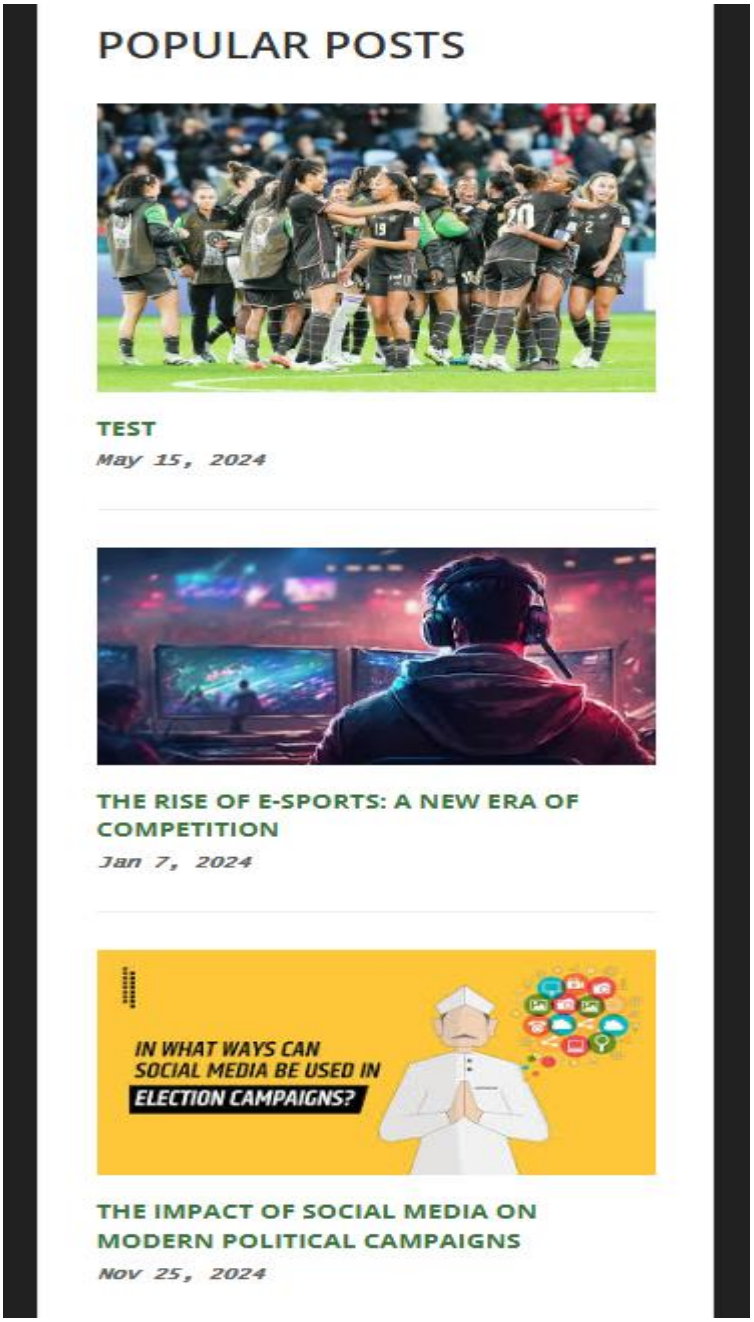

Рисунок 4.27 – Список популярних статей на блозі

Джерело: розроблено автором

Пагінація сторінки працює відмінно. Статті поділені на 3 сторінки. Пагінація статей зображена на рисунку 4.28

Рисунок 4.28 – Пагінація сторінок

Джерело: розроблено автором

Також є можливість переглянути категорії статей, які доступні і їх кількість. Список категорій зображено на рисунку 4.29

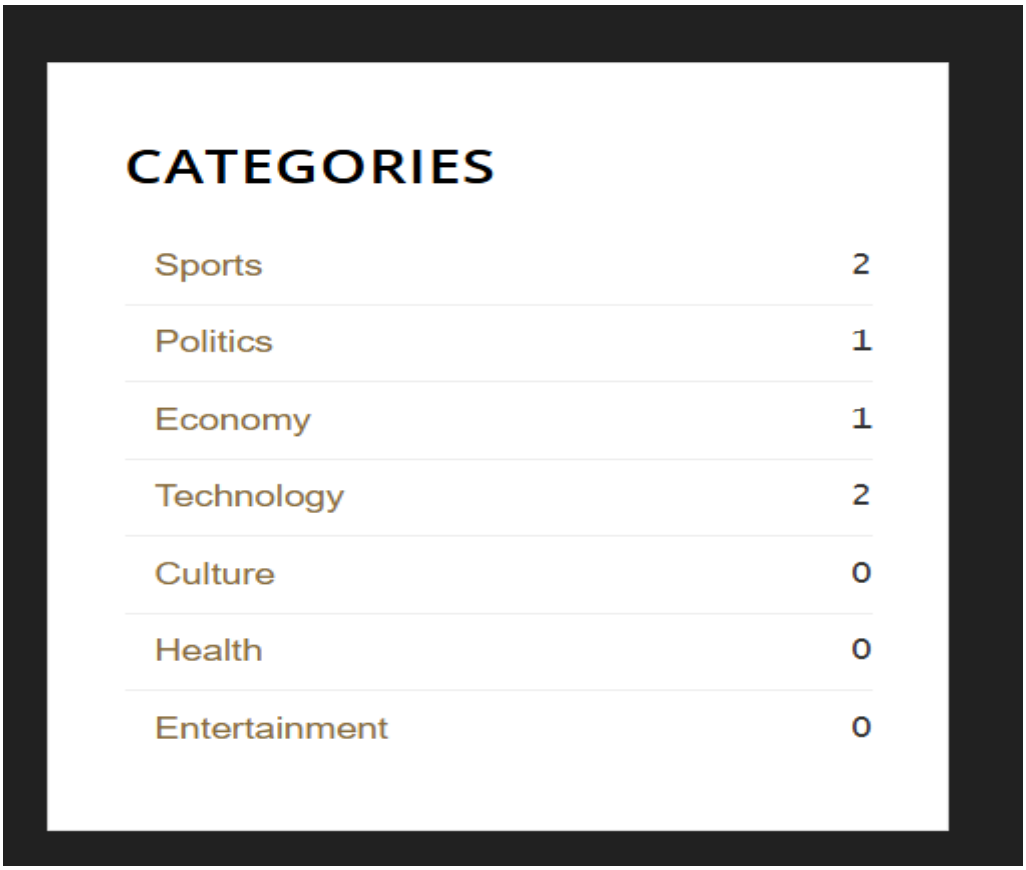

Рисунок 4.29 – Список категорій на блозі

Джерело: розроблено автором

Вікно окремої відкритої статті зображено на рисунку 4.30

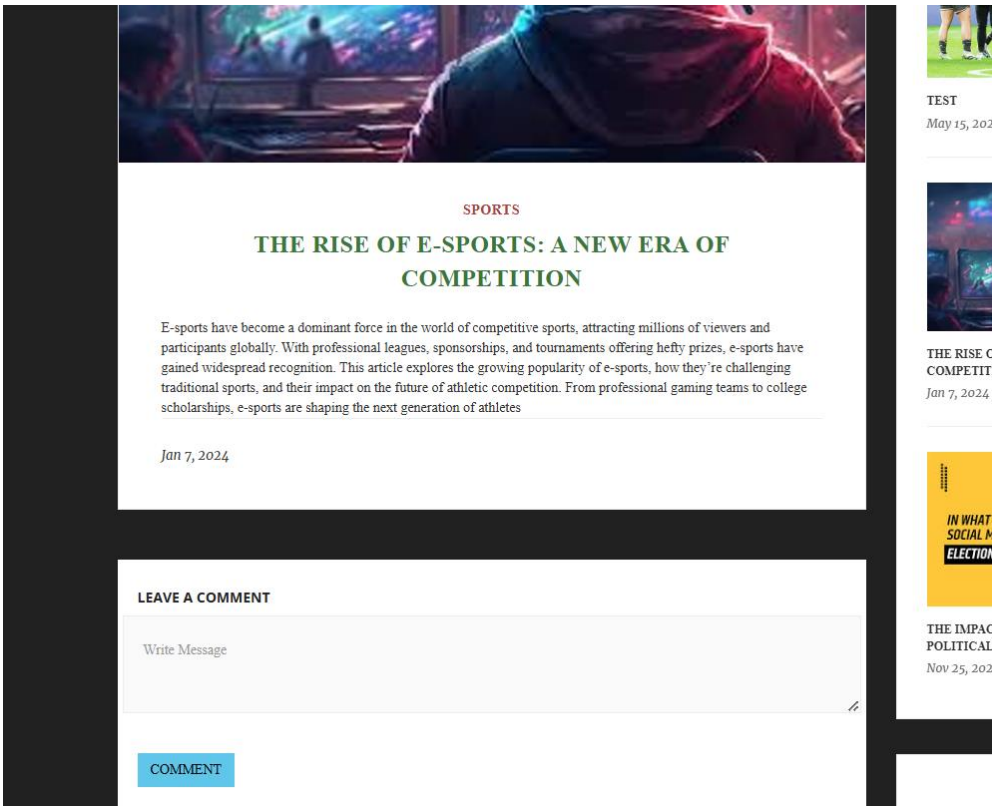

Рисунок 4.30 – Вікно окремої відкритої сторінки статті

Джерело: розроблено автором

Маємо можливість залишити коментар до статті. Для цього пишемо сам коментар у відповідне поле і натискаємо Post Comment. Процес додавання коментаря до статті зображено на рисунку 4.31

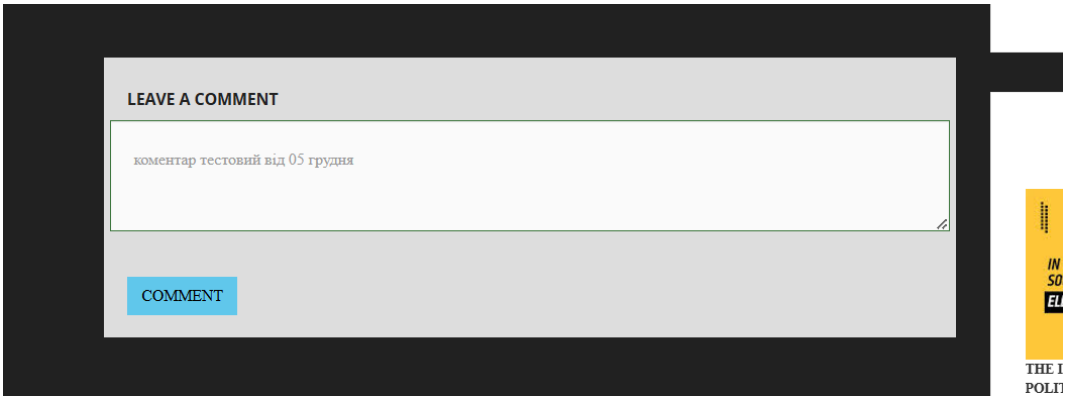

Рисунок 4.31 – Процес додавання коментаря до статті

Джерело: розроблено автором

Результат успішного додавання коментаря до статті зображено на рисунку 4.32

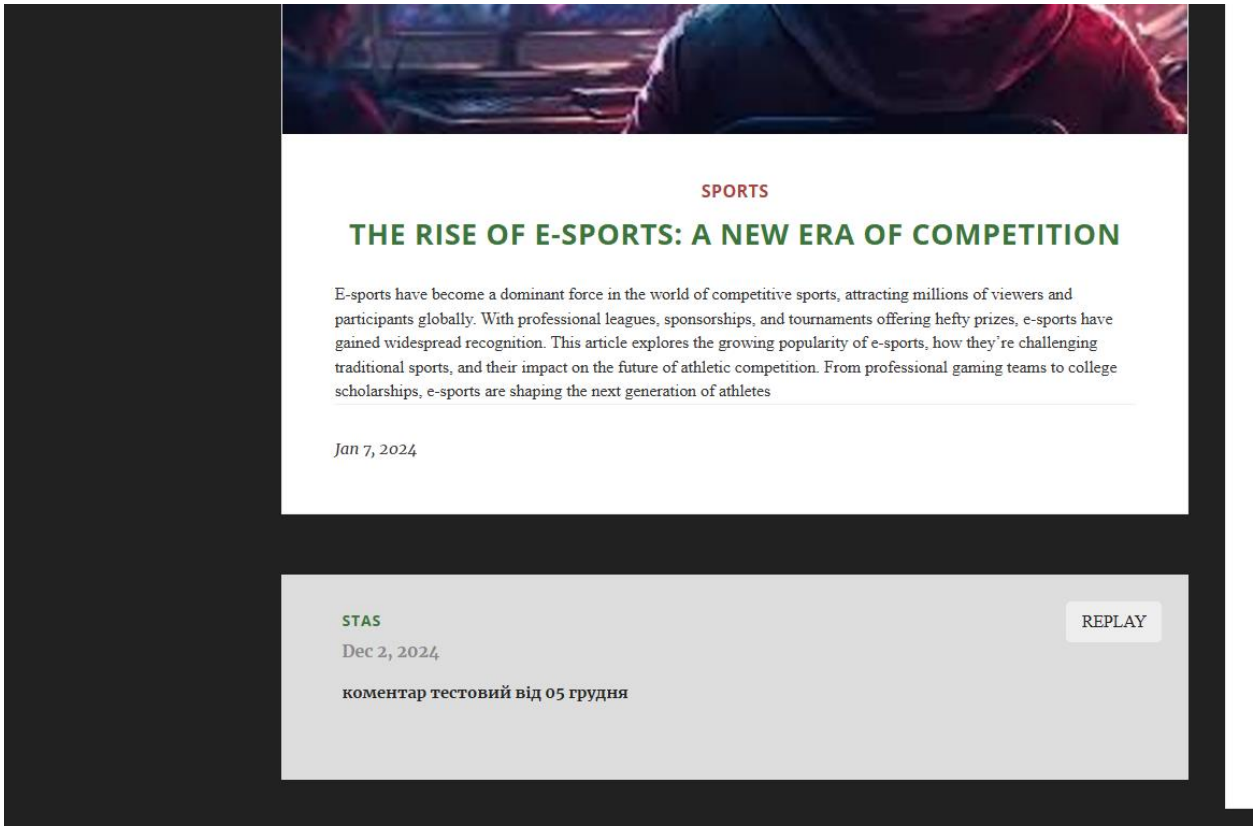

Рисунок 4.32 – Результат успішного додавання коментаря до статті

Джерело: розроблено автором

Список статей за конкретною категорією представлено на рисунку 4.33

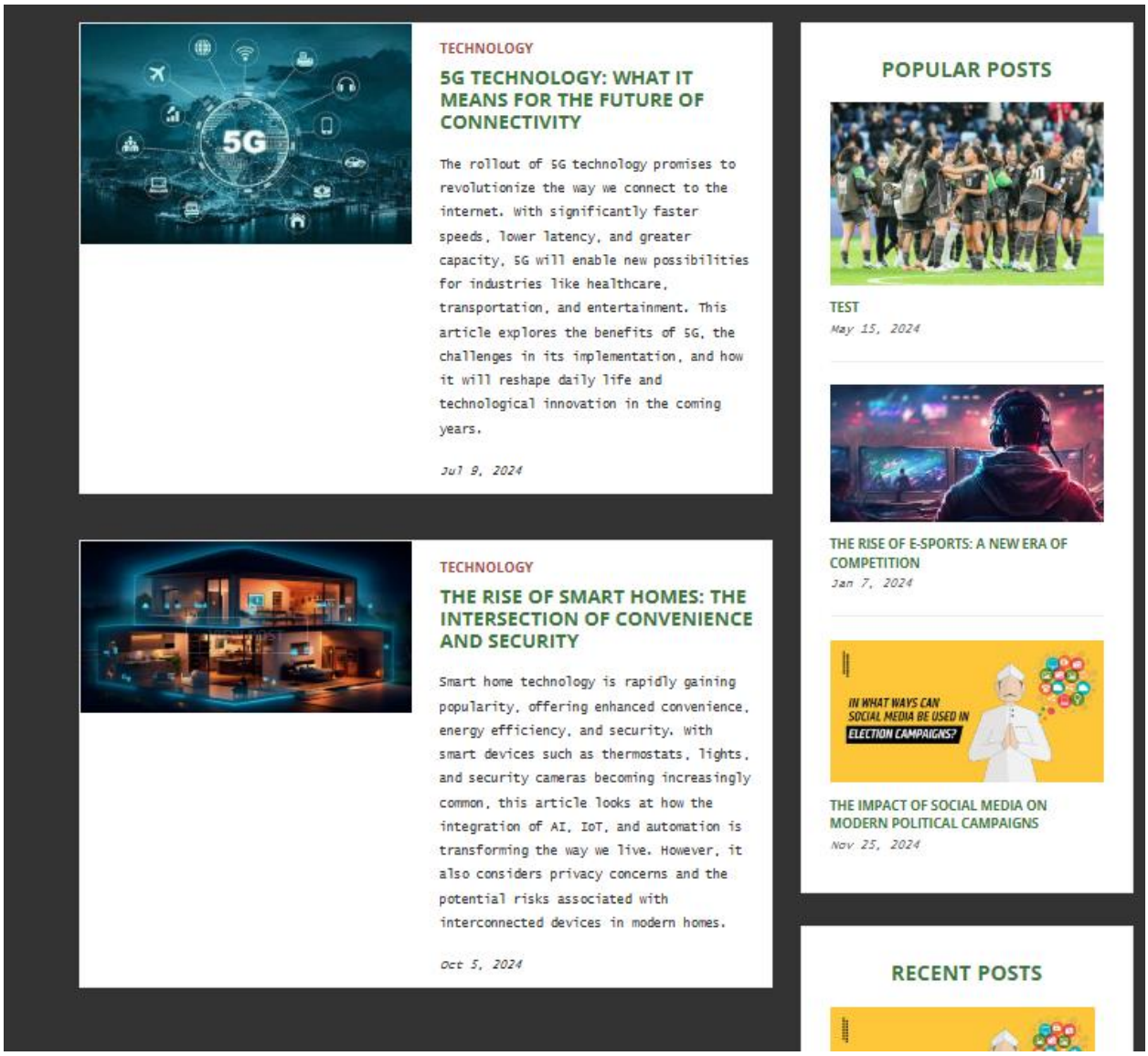

Рисунок 4.33 – Список статей за конкретною категорією

Джерело: розроблено автором

ВИСНОВКИ

У результаті виконання даної кваліфікаційної роботи магістра було створено веб-орієнтовану систему підтримки діяльності новинного блогу.

На початку було досліджено актуальність проблеми. Провівся огляд літературних джерел, які були близькими до поставленої мети дипломного проектування. Спираючись на проведений аналіз предметної області була сформована мета дослідження, яка відповідає тематиці кваліфікаційної роботи магістра та надає деталізований опис задач, які необхідно вирішити для досягнення мети.

З метою побудови чітких вимог до майбутнього продукту було проведено аналіз програмних продуктів-аналогів, таких як «Blogger» та «LiveJournal». По кожному додатку-аналогу представлено переваги та недоліки їх використання для розробки веб орієнтованої системи. По результатам аналізу було прийнято рішення розробити власну веб орієнтовану систему, яка дозволить усунути наявні недоліки та додати необхідний функціонал. Представлено постановку задачі і описано кроки, які потрібно виконати щоб досягти мети розробки.

З метою визначення засобів реалізації, проведено дослідження фреймворків React та Angular, які можуть бути застосовані для розробки новинного блогу. Проведено порівняльний аналіз вказаних фреймворків із зазначенням переваг та недоліків кожного. За результатами порівняння з урахуванням визначених завдань даної роботи оптимальним вибором є php-фреймворк yii2.

Виконано функціональне моделювання веб орієнтованої системи в IDEF0. Також виконано декомпозицію контексту на першому рівні моделювання за допомогою нотації IDEF0, що полягає у представленні складного процесу в окремих компонентах. Представлено діаграму варіантів використання, яка відображає поведінку системи в нотації UML. Проведено

моделювання, описано фізичну структуру бази даних. Було представлено архітектуру веб додатку та шлях реалізації бази даних.

Виконана програмна реалізація веб-орієнтованої системи підтримки діяльності новинного блогу, що базується на обраній технології PHP-фреймворка Yii2. В рамках реалізації було розроблено інтерфейс користувача, систему управління контентом, а також функціонал для адміністрування та взаємодії з користувачами. Реалізовано механізми створення нових дописів, коментування та категоризації контенту, що значно покращує зручність та ефективність роботи з блогом. Особлива увага була приділена безпеці даних і оптимізації продуктивності системи. У результаті виконаної роботи створено додаток, що відповідає визначеним вимогам і забезпечує зручну підтримку діяльності новинного блогу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Як провести аналіз предметної галузі у дипломній роботі [Електронний ресурс] – Доступ до ресурсу: https://na5ku.com.ua/uk/yak-provesti-analiz-predmetnoi-galuzi-u-diplomnij-roboti/ (Дата звернення: 15.11.2024)

2. Що таке Блог і для чого він потрібний сайту [Електронний ресурс] – Доступ до ресурсу: https://voicemarketing.com.ua/blog/shcho-take-blog-i-dlya-choho-vin-potribnyy-saytu/ (Дата звернення: 09.11.2024)

3. Блоги: визначення, типи, переваги та недоліки [Електронний ресурс] – Доступ до ресурсу: https://uk.blogpascher.com/%D0%9A%D0%B5%D1%80%D1%96%D0%B2%D0%BD%D0%B8%D1%86%D1%82%D0%B2%D0%B0/%D0%91%D0%BB%D0%BE%D0%B3 (Дата звернення: 05.11.2024)

4. How to start a news blog and start breaking perspectives [Електронний ресурс] – Доступ до ресурсу: https://getlasso.co/how-to-start-a-blog/news/ (Дата звернення: 05.11.24)

5. 10 порад, як створити класний блог самостійно [Електронний ресурс] – Доступ до ресурсу: https://infounion.com.ua/ua/blog/kak-sozdat-blog-samostojatelno.html (Дата звернення: 10.11.2024)

6. Аналіз конкурентів з погляду функціоналу сайту та асортименту [Електронний ресурс] – Доступ до ресурсу: https://elit-web.ua/ua/blog/analiz-konkurentov-s-tochki-zreniya-funkcionala (Дата звернення: 03.11.24)

7. Найкращі платформи для ведення блогу [Електронний ресурс] – Доступ до ресурсу: https://prposting.com/uk/blog/29-blogging-platforms (Дата звернення: 04.11.2024)

8. Create a unique and beautiful blog easily [Електронний ресурс] – Доступ до ресурсу: https://www.blogger.com/about/ (Дата звернення: 12.11.24)

9. 10 найкращих платформ для блогів для творців контенту [Електронний ресурс] – Доступ до ресурсу: https://www.doola.com/uk/blog/top-10-blogging-platforms-for-content-creators/ (Дата звернення: 10.11.2024)

10. Blogger - огляд сервіса [Електронний ресурс] – Доступ до ресурсу: https://vchymo.com/app/application/Blogger (Дата звернення: 06.11.2024)

11. LiveJournal: Discover global communities of bloggers [Електронний ресурс] – Доступ до ресурсу: https://www.livejournal.com/ (Дата звернення: 12.11.24)

12. Which platform to choose for a blog [Електронний ресурс] – Доступ до ресурсу: https://cityhost.ua/en/blog/kakuyu-platformu-vybrat-dlya-bloga.html (Дата звернення: 06.11.2024)

13. Методичні вказівки до переддипломної практики для студентів освітнього ступеня «магістр» [Електронний ресурс] – Доступ до ресурсу: https://mix.sumdu.edu.ua/textbooks/83235/2226752/MV_pereddyplomna_praktyka_mahistriv.pdf#page=17&zoom=100,53,445 (Дата звернення: 01.11.24)

14. Повний посібник з Yii 2.0 [Електронний ресурс] – Доступ до ресурсу: https://www.yiiframework.com/doc/guide/2.0/uk/start-gii (Дата звернення: 03.11.2024)

15. База даних MySQL [Електронний ресурс] – Доступ до ресурсу: https://promoter.net.ua/articles/baza-danix-mysql.html (Дата звернення: 10.11.2024)

16. Що таке Sitemap.xml чи карта сайту [Електронний ресурс] – Доступ до ресурсу: https://elit-web.ua/ua/blog/sitemap#:~:text=%D0%9A%D0%B0%D1%80%D1%82%D0%B0%20%D1%81%D0%B0%D0%B9%D1%82%D1%83%20(%D0%B0%D0%B1%D0%BE%20sitemap)%20%E2%80%93,%D1%81%D1%82%D0%BE%D1%80%D1%96%D0%BD%D0%BA%D0%B8%20%D1%82%D0%B0%20%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B8%20%D0%B2%D0%B5%D0%B1%2D%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81%D1%83 (Дата звернення: 15.11.24)

17. Топ-5 фреймворків для Front-end розробки [Електронний ресурс] – Доступ до ресурсу: https://wezom.academy/ua/top-5-frejmvorkov-dlja-front-end-razrabotki/ (Дата звернення: 16.11.24)

18. React vs Angular: Which JS Framework to choose for Front-end Development? [Електронний ресурс] – Доступ до ресурсу: https://radixweb.com/blog/react-vs-angular (Дата звернення: 13.11.24)

19. Angular vs React: Which to Choose for Your Front End in 2024? [Електронний ресурс] – Доступ до ресурсу: https://www.simform.com/blog/angular-vs-react/ (Дата звернення: 14.11.24

20. Angular проти React [Електронний ресурс] – Доступ до ресурсу: https://foxminded.ua/angular-vs-react/ (Дата звернення: 12.11.24)

21. Робота з фреймворком Yii2: Швидкість розробки та ефективність [Електронний ресурс] – Доступ до ресурсу: https://promoter.net.ua/articles/robota-z-freimvorkom-yii2-svidkist-rozrobki-ta-efektivnist.html (Дата звернення: 15.11.24)

22. The complete guide to understand IDEF diagram [Електронний ресурс] – Доступ до ресурсу: https://www.edrawmax.com/article/the-complete-guide-to-understand-idef-diagram.html (Дата звернення: 03.11.24)

23. Створення схем IDEF0 [Електронний ресурс] – Доступ до ресурсу: https://support.microsoft.com/uk-ua/topic/%D1%81%D1%82%D0%B2%D0%BE%D1%80%D0%B5%D0%BD%D0%BD%D1%8F-%D1%81%D1%85%D0%B5%D0%BC-idef0-ea7a9289-96e0-4df8-bb26-a62ea86417fc (Дата звернення: 08.11.2024)

24. Що таке діаграма варіантів використання UML: символи, шаблони, інструмент і посібник [Електронний ресурс] – Доступ до ресурсу: https://www.mindonmap.com/uk/blog/what-is-a-uml-use-case-diagram/#part1 (Дата звернення: 02.11.24)

25. Фізичне проектування структури бази даних [Електронний ресурс] – Доступ до ресурсу: https://rdb.dp.ua/uk/chapter_04 (Дата звернення: 09.11.2024)Angular vs React: Which to Choose for Your Front End in 2024?

[Електронний ресурс] – Доступ до ресурсу: https://www.simform.com/blog/angular-vs-react/ (Дата звернення: 14.11.24)

26. Модель-Представлення-Контролер (MVC) [Електронний ресурс] – Доступ до ресурсу: https://www.yiiframework.com/doc/guide/1.1/uk/basics.mvc (Дата звернення: 14.11.24)

27. Database Migration. Work with database // Yiiframework [Електронний ресурс] – Доступ до ресурсу: https://www.yiiframework.com/doc/guide/2.0/uk/db-migrations (Дата звернення: 15.11.24)

28. Генерування коду за допомогою Gii. Перше знайомство // Yiiframework [Електронний ресурс] – Доступ до ресурсу: https://www.yiiframework.com/doc/guide/2.0/uk/start-gii (Дата звернення: 16.11.2024)

29. Pagination. Displaying Data // Yiiframework [Електронний ресурс] – Доступ до ресурсу: https://www.yiiframework.com/doc/guide/2.0/en/output-pagination (Дата звернення: 14.11.2024)

30. Етапи управління IT-проєктами: ініціація проєкту для менеджера [Електронний ресурс] – Доступ до ресурсу: https://iampm.club/ua/blog/etapi-upravlinnya-it-projektami-inicziacziya-projektu-dlya-menedzhera/ (Дата звернення: 11.11.2024)

31. How to write SMART goals [Електронний ресурс] – Доступ до ресурсу: https://www.atlassian.com/blog/productivity/how-to-write-smart-goals (Дата звернення: 08.11.2024)

32. Що таке SMART-цілі? [Електронний ресурс] – Доступ до ресурсу: https://experience.dropbox.com/uk-ua/resources/smart-goals (Дата звернення: 03.11.24)

33. What is Work Breakdown Structure (WBS) diagram? [Електронний ресурс] – Доступ до ресурсу: https://www.edrawsoft.com/what-is-work-breakdown-structure-diagram.html (Дата звернення: 02.11.24)

34. Структура розбиття робіт (Work Breakdown Structure – WBS) [Електронний ресурс] – Доступ до ресурсу: https://www.maxzosim.com/struktura-rozbittia-robit/ (Дата звернення: 04.11.24)

35. Що таке діаграма Ганта та як нею правильно користуватися? [Електронний ресурс] – Доступ до ресурсу: https://nachasi.com/creative/2020/09/03/gantt-chart/ (Дата звернення: 03.11.24)

ДОДАТОК А
ПЛАНУВАННЯ РОБІТ

А.1 Ідентифікація мети ІТ-проекту

Управління ІТ-проектами є складним процесом. Він включає чітко структурований процес, який в свою чергу ділиться на етапи. Цей процес точно не можна назвати набором випадкових дій. Кожна дія із цих етапів має свою мету та ціль.[30]

Метою цього ІТ-проекту є розробка веб-орієнтованої системи для створення та управління новинним блогом, що дозволяє користувачам переглядати, додавати коментарі, а також адмініструвати сайт через зручну панель управління. Ця система повинна забезпечити ефективне управління контентом, зручність для користувачів та адміністраторів, а також можливість розширення функціоналу в майбутньому. Ось конкретні цілі проекту:

1. Розробка бази даних: Створити чотири основні таблиці для зберігання даних користувачів, статей, категорій і коментарів. Це дозволить зберігати всю необхідну інформацію для роботи блогу та забезпечить надійне зберігання даних.

2. Реалізація функціоналу для користувачів: Надати можливість користувачам реєструватися, авторизуватися, переглядати статті, додавати коментарі до публікацій та взаємодіяти з контентом через інтерфейс блогу.

3. Реалізація адміністративної панелі: Створити систему для адміністраторів, що дозволяє керувати статтями, категоріями, користувачами та коментарями. Адміністратор повинен мати можливість створювати, редагувати та видаляти контент, а також переглядати статистику переглядів статей.

4. Розширена функціональність: Впровадити можливість динамічного оновлення кількості переглядів статей, додавання зображень до

статей, категоризації контенту, а також реалізувати пагінацію для відображення статей.

5. Підвищення зручності використання: Забезпечити зручний інтерфейс як для звичайних користувачів, так і для адміністраторів, реалізувати механізми для комфортного взаємодії з контентом і управління ним, зокрема через систему тегів, категорій та рейтингу.

6. Інтеграція системи пошуку: Розробити механізм для пошуку статей за тегами, категоріями та іншими характеристиками, що підвищить зручність для користувачів при навігації по блогу.

Метод SMART означає Specific, Measurable, Achievable, Relevant і Time-Bound. Оцінка цих параметрів у контексті цілей IT-проекту сприяє впевненості в тому, що завдання можна виконати в межах встановленого часу. Такий підхід усуває нечіткість і припущення, визначає конкретний термін виконання та спрощує моніторинг розвитку і виявлення пропущених етапів. [31]

Результат деталізації даного проекту методом SMART розміщено у таблиці А.1.

Таблиця А.1 – Деталізація мети проекту методом SMART

Specific (конкретна)	Створення веб-додатку для підтримки діяльності новинного блогу та управління контентом.
Measurable (вимірювана)	Успіх проекту буде оцінюватися за кількістю зареєстрованих користувачів, створених статей, коментарів та переглядів
Achievable (досяжна, узгоджена)	Мета досяжна завдяки чітко визначеному технічному завданню, використанню сучасних фреймворків і доступу до необхідних ресурсів для розробки та тестування.
Relevant (актуальна)	Актуальність пов'язана з попитом на прості платформи для ведення блогів, які дозволяють користувачам легко взаємодіяти з контентом і керувати ним, а також забезпечують зручний інтерфейс для адміністраторів.
Time-framed (обмежена в часі)	Повністю завершити розробку, провести тестування та впровадити продукт до 09.12.2024

Джерело: розроблено автором

А.2 Планування змісту структури робіт ІТ-проекту

Детальний опис етапів створення веб-орієнтованої інформаційної системи наведено в таблиці А.1.

Таблиця А.1 – Етапи створення веб-орієнтованої інформаційної системи

	Склад і зміст робіт	Строк розробки (у робочих днях)
1	Аналіз предметної області	13 днів
2	Складання технічного завдання	9 днів
3	Підготовка прототипу	12 днів
4	Створення макету дизайну	3 дні
5	Розробка модулю реєстрації/авторизації користувачів	5 днів
6	Розробка модулю додавання/редагування/видалення статей	2 дні
7	Розробка модулю додавання/видалення категорій	1 день
8	Розробка динамічної зміни кількості переглядів кожної статті	1 день
9	Розробка модулю додавання коментарів від користувачів	3 дні
10	Розробка модулю відображення підбірки найпопулярніших та нещодавно доданих статей	2 дні
11	Створення пагінації на головній сторінці	1 день
12	Розробка модулю відображення статей по категоріям та їх кількості.	2 дні
13	Beta-тестування	5 днів
14	Alpha-тестування	4 дні
15	Перевірка працездатності системи	2 дні
16	Реліз системи	1 день
17	Створення програмної документації	5 днів
	Загальна тривалість робіт	71 день

Джерело: розроблено автором

Веб-додатки надають великий спектр функцій і є оптимальним варіантом для вирішення різних завдань.

Розробка програми повинна бути завершена в межах запланованого часу та з використанням доступних ресурсів. [32]

А.3 Побудова календарного графіку виконання ІТ - проекту

Діаграма WBS зазвичай розбиває основну мету проекту на менші та більш керовані компоненти (пакети робіт), призначені для конкретних відділів, які мають виконати певні завдання. Пакет робіт — це результат найнижчого рівня структури розподілу робіт, який використовують учасники команди для виконання конкретних завдань. [33]

Правильно розроблена WBS дає змогу чітко віднести кожну активність проекту до одного і тільки одного кінцевого елемента цієї структури. Окрім функції обліку витрат, WBS також сприяє перекладу вимог з одного рівня специфікацій системи на інший.

WBS може бути представленою горизонтально у вигляді схеми або вертикально, як деревоподібна структура (аналогічно організаційній діаграмі).

Зазвичай створення WBS відбувається на початкових етапах проекту і передує більш детальному плануванню завдань та етапів реалізації. [34]

Діаграму WBS представлено на рисунку А.1

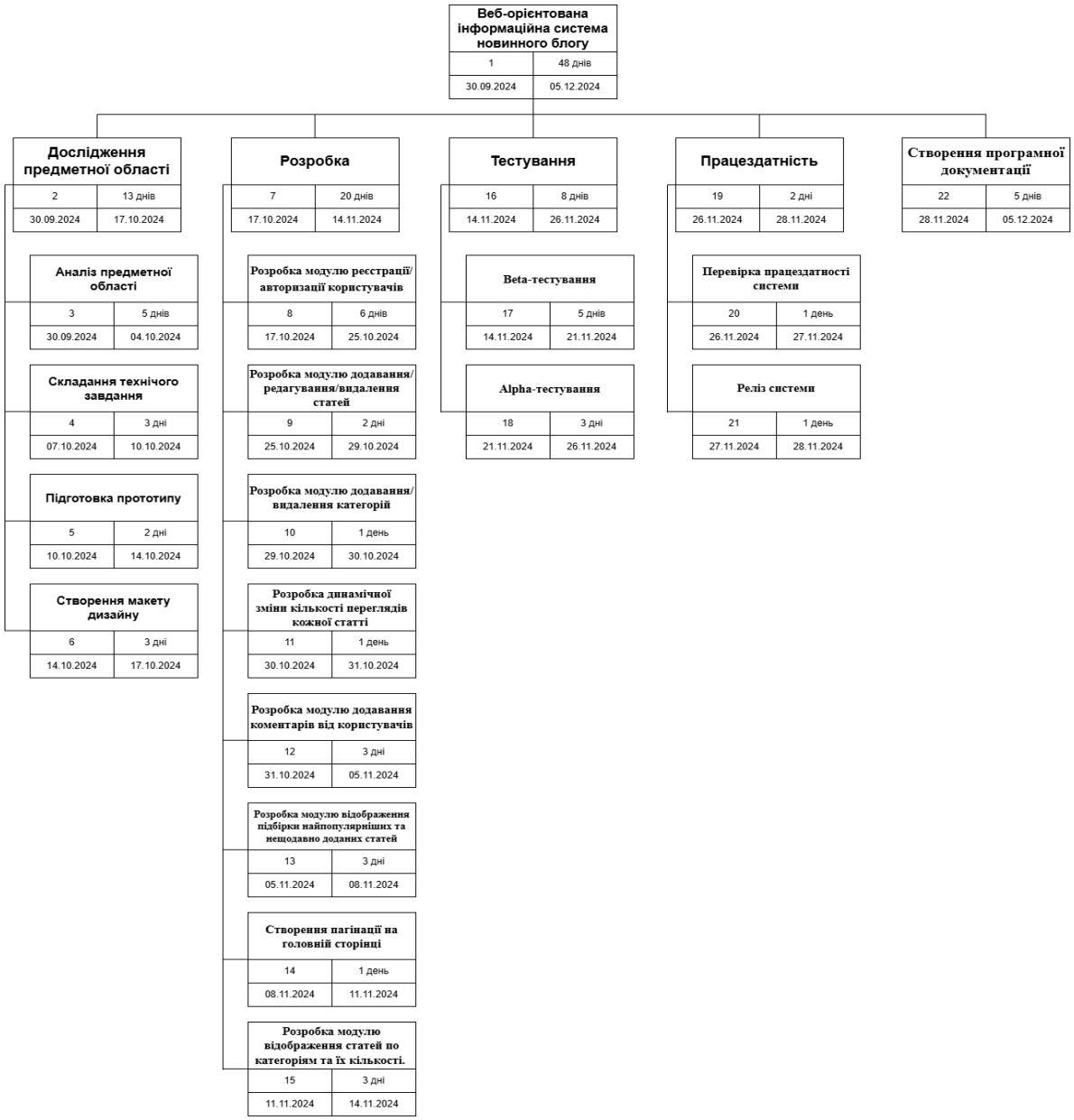

Рисунок А.1 – Планування змісту структури роботи за допомогою діаграми WBS

Джерело: розроблено автором

Після поділу процесів на етапи наступним кроком є розробка графічної схеми, яка відображає учасників або осіб, відповідальних за виконання завдань у проекті. Така схема називається організаційною структурою виконання (OBS). Відповідальні особи в цій структурі — це працівники, які координують і реалізують завдання, визначені у WBS. Кожне з цих завдань можна трактувати як окремий проект.

На рисунку А.2 представлена організаційна структура для планування проекту.

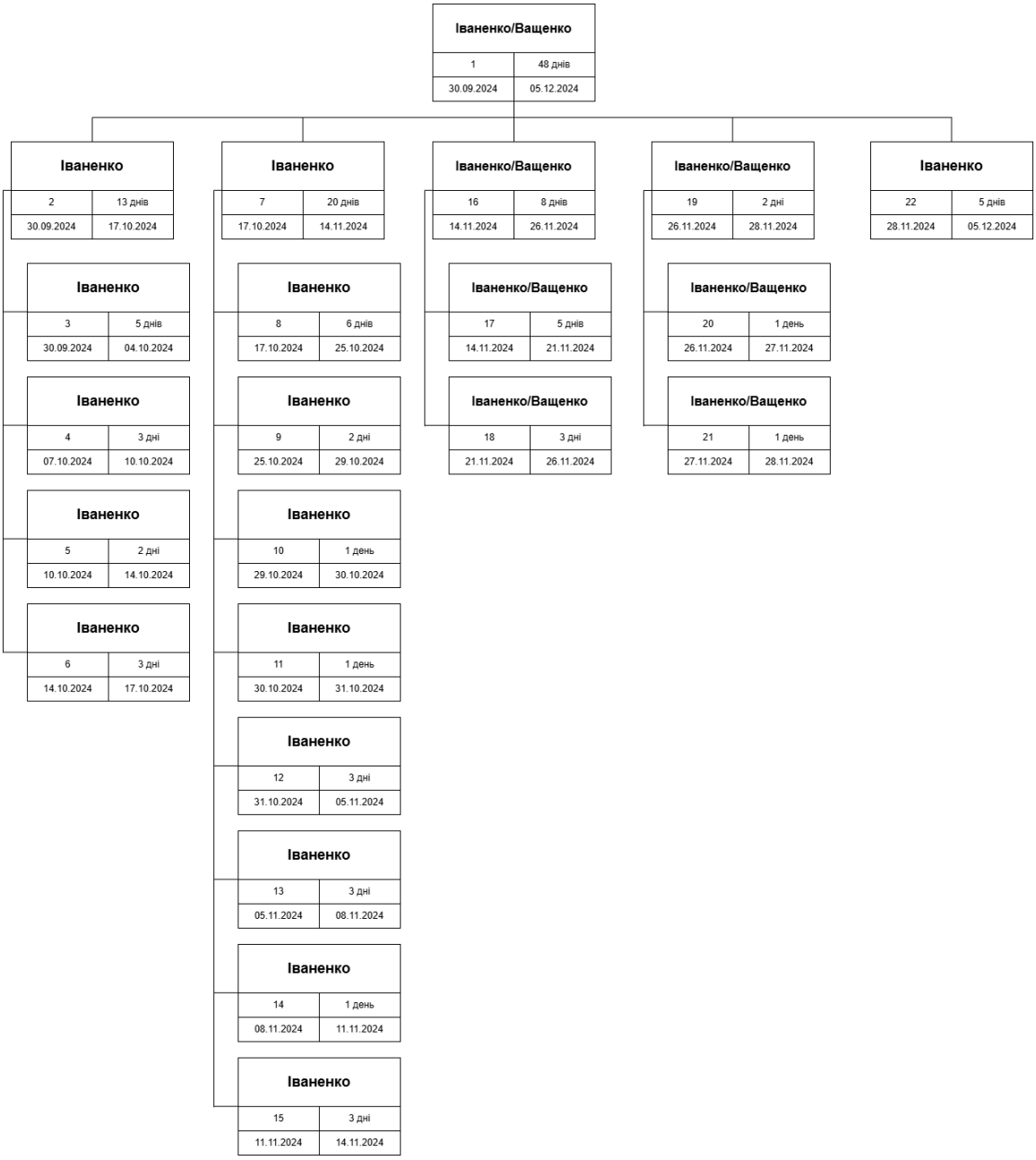

Рисунок А.2 – OBS-структура проекту

Джерело: розроблено автором

Діаграма Ганта – це графічне зображення переліку завдань за часом. Вісь X показує хронологію, а вісь Y – задачі, які потрібно виконати. Однак Гант не був першим, хто запропонував організовувати й планувати хід проєкту у вигляді послідовних стовпців, розташованих один за одним.. [35]

Календарний графік проекту можна побачити на рисунках А.3-А.5.

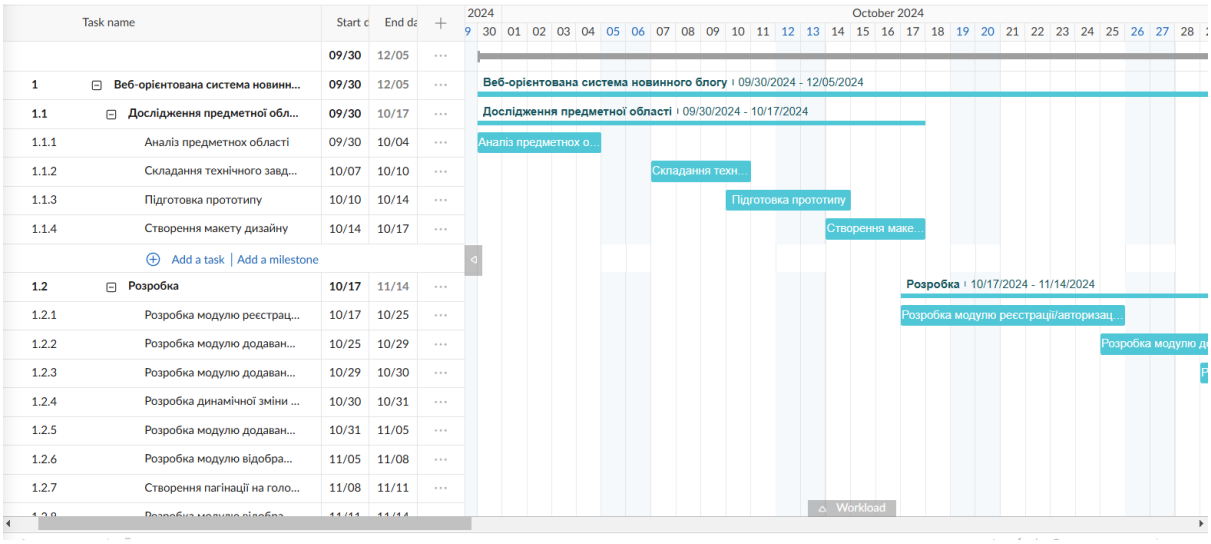

Рисунок А.3 – Загальний вигляд діаграми всього проекту, частина 1

Джерело: розроблено автором

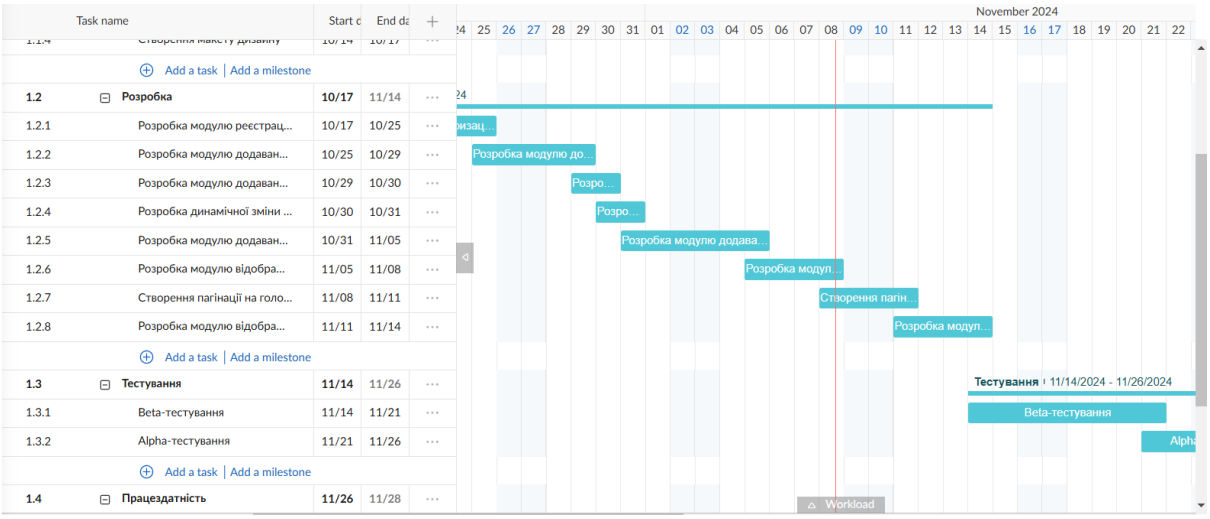

Рисунок А.4 – Загальний вигляд діаграми всього проекту, частина 2

Джерело: розроблено автором

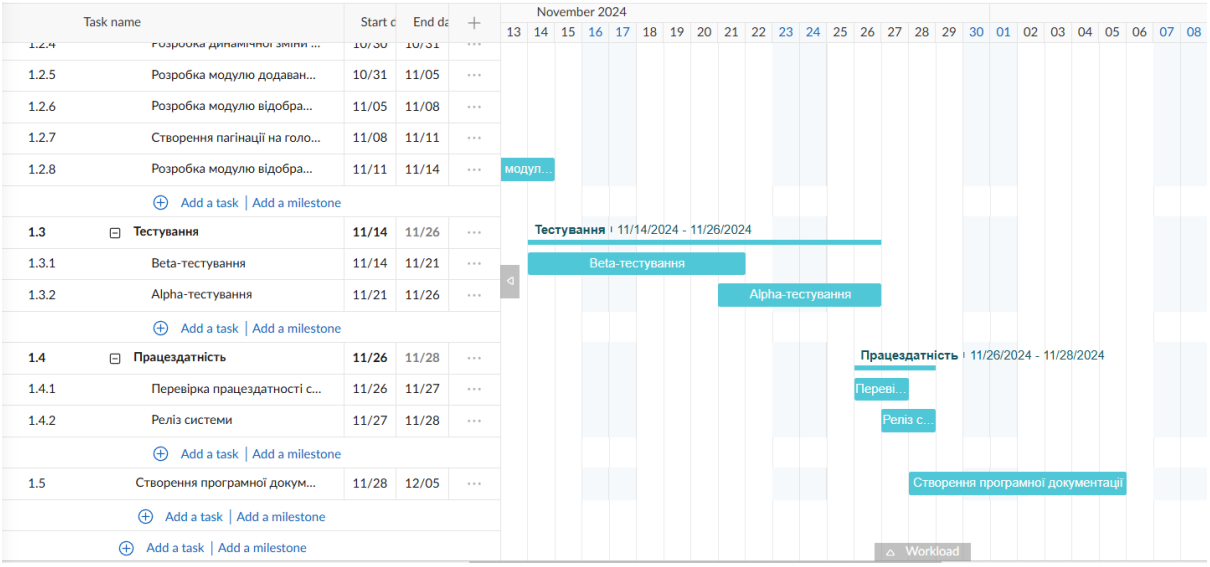

Рисунок А.5 – Загальний вигляд діаграми всього проекту, частина 3

Джерело: розроблено автором

А.4 Планування ризиків проекту

Шкала класифікації ризиків за їх впливом на проект та ймовірністю виникнення подана в таблиці А.3.

Таблиця А.3 – Шкала класифікації ризиків за ймовірністю виникнення та величиною впливу на проект

Оцінка	Ймовірність виникнення	Вплив ризику	Тип ризику
1	Низька	Низький	Прийнятні
2	Середня	Середній	Виправдані
3	Висока	Високий	Недопустимі

Джерело: розроблено автором

Для мінімізації негативного впливу ризиків на проєкт необхідно розробити план дій щодо їхнього усунення. Це включає оцінку результативності запропонованих заходів та аналіз можливих наслідків, які

можуть вплинути на реалізацію проєкту. Оцінювання здійснюється за критеріями, наведеними в таблиці А.3. У процесі планування була створена матриця ймовірності виникнення ризиків і їхнього впливу, яка представлена в таблицях А.4-А.5. Ризики, позначені зеленим кольором, вважаються прийнятними, жовтим — обґрунтованими, а червоним — неприпустимими.

Таблиця А.4 – Матриця ймовірності та впливу

Ймовірність ризику(Й)	Вплив загрози(ризику)				
	Дуже малий	Малий	Середній	Великий	Дуже великий
	0,05	0,1	0,2	0,4	0,8
0,9					
0,7					
0,5			R2(0,1), R6(0,1)		R1(0,4)
0,3			R4(0,06), R8(0,06)		
0,1			R3(0,02), R7(0,02)	R5(0,04)	

Джерело: розроблено автором

Таблиця А.5 містить класифікацію ризиків за рівнем, відповідно до отриманого значення індексу. У таблицях А.6.-А.7. надано опис ризиків та відповідні стратегії реагування на кожен з них.

Таблиця А.5 – Шкала оцінювання за рівнем ризику

№	Назва	Межі	Ризики, які входять(номера)
1	Прийнятні	$0,005 \leq R \leq 0,05$	3,5,7
2	Виправдані	$0,05 < R \leq 0,14$	2,4,6,8
3	Недопустимі	$0,14 < R \leq 0,72$	1

Джерело: розроблено автором

Таблиця А.6 – Стратегії реагування на ризики

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив	Ранг ризику	План А	Тип стратегії	План Б
RS_1	Відкритий	Затримки через недостатню кількість ресурсів (розробники, час)	Середній	Дуже великий	0,4	Отримати знання по ходу виконання	Ухилення	Негайна консультація з викладачем
RS_2	Відкритий	Погіршення продуктивності через погано оптимізований код	Середній	Середній	0,1	Регулярно проводити профілювання продуктивності та оптимізувати код.	Ухилення	

RS_3	Відкритий	Виникнення багів після оновлення або змін в коді	Малий	Середній	0,02	Тестувати всі зміни в коді на локальному середовищі та використовувати систему контролю версій.	Зменшення	Використовувати автоматичні тести і забезпечити наявність резервних копій для швидкого відновлення
RS_4	Відкритий	Відсутність інтернет-з'єднання	Малий	Середній	0,06	Використовувати мобільний інтернет або резервний канал зв'язку	Попередження	Використати техніку в іншому місці
RS_5	Відкритий	Втрата або пошкодження даних	Малий	Великий	0,04	Регулярно створювати резервні копії даних	Ухилення	Використовувати хмарні сервіси

Таблиця А.7 – Продовження таблиці А.6

RS_6	Відкритий	Проблеми з комунікацією в команді	Середній	Середній	0,1	Використання інструментів для командної роботи	Попередження	Призначити регулярні наради для уточнення статусу завдань
RS_7	Відкритий	Зміна вимог ТЗ	Малий	Середній	0,02	Обговорити додаткові терміни для виконання	Зменшення	Зробити відповідні зміни в ТЗ
RS_8	Відкритий	Хвороба ключового співробітника	Середній	Середній	0,06	Залучення тимчасових працівників	Зменшення	Пришвидшити роботу за рахунок зменшення розробки додаткових функцій

Джерело: розроблено автором

ДОДАТОК Б

Основний програмний код

SiteController:

```
<?php

namespace app\controllers;

use app\models\Article;
use app\models\Category;
use app\models\CommentForm;
use Yii;
use yii\data\Pagination;
use yii\filters\AccessControl;
use yii\web\Controller;
use yii\filters\VerbFilter;
use app\models\LoginForm;
use app\models\ContactForm;


class SiteController extends Controller
{
    /**
     * {@inheritdoc}
     */
    public function behaviors()
    {
        return [
```

```
        'access' => [
            'class' => AccessControl::class,
            'only' => ['logout'],
            'rules' => [
                [
                    'actions' => ['logout'],
                    'allow' => true,
                    'roles' => ['@'],
                ],
            ],
        ],
        'verbs' => [
            'class' => VerbFilter::class,
            'actions' => [
                'logout' => ['post'],
            ],
        ],
    ];
}

/**
 * {@inheritdoc}
 */
public function actions()
{
    return [
        'error' => [
            'class' => 'yii\web\ErrorAction',
        ],
        'captcha' => [
            'class' => 'yii\captcha\CaptchaAction',
            'fixedVerifyCode' => YII_ENV_TEST ? 'testme' : null,
        ],
```

```
        ];
    }

    /**
     * Displays homepage.
     *
     * @return string
     */
    public function actionIndex()
    {
        $data = Article::getAll(2);

        $popular = Article::getPopular();
        $recent = Article::getRecent();
        $categories = Category::getAll();

        return $this->render('index', [
            'articles'=>$data['articles'],
            'pagination'=>$data['pagination'],
            'popular'=>$popular,
            'recent'=>$recent,
            'categories'=>$categories
        ]);
    }

    public function actionView($id)
    {
        $article = Article::findOne($id);
        $popular = Article::getPopular();
        $recent = Article::getRecent();
        $categories = Category::getAll();
        $comments = $article->comments;
        $commentForm = new CommentForm();
```

```
        $article->viewedCounter();

        return $this->render('singleArticle', [
            'article'=>$article,
            'popular'=>$popular,
            'recent'=>$recent,
            'categories'=>$categories,
            'comments'=>$comments,
            'commentForm'=>$commentForm
        ]);
    }

    public function actionCategory($id)
    {
        $data = Category::getArticlesByCategory($id);
        $popular = Article::getPopular();
        $recent = Article::getRecent();
        $categories = Category::getAll();

        return $this->render('category', [
            'articles'=>$data['articles'],
            'pagination'=>$data['pagination'],
            'popular'=>$popular,
            'recent'=>$recent,
            'categories'=>$categories
        ]);
    }



    /**
     * Displays contact page.
     *
```

```
 * @return Response|string
 */
public function actionContact()
{
    $model = new ContactForm();
    if ($model->load(Yii::$app->request->post()) && $model->contact(Yii::$app->params['adminEmail'])) {
        Yii::$app->session->setFlash('contactFormSubmitted');

        return $this->refresh();
    }
    return $this->render('contact', [
        'model' => $model,
    ]);
}

public function actionComment($id)
{
    $model = new CommentForm();

    if(Yii::$app->request->isPost)
    {
        $model->load(Yii::$app->request->post());
        if($model->saveComment($id))
        {
            Yii::$app->getSession()->setFlash('comment', 'Your comment will be added soon!');
            return $this->redirect(['site/view','id'=>$id]);
        }
    }
}

/**
 * Displays about page.
 *
```

```
     * @return string
     */
    public function actionAbout()
    {
        return $this->render('about');
    }
}
```

AuthController.php:

```
<?php

namespace app\controllers;

use app\models\LoginForm;
use app\models\SignupForm;
use app\models\User;
use yii\web\Controller;
use yii\web\Response;
use Yii;

class AuthController extends Controller
{
    /**
     * Login action.
     *
     * @return Response|string
     */
    public function actionLogin()
    {
        if (!Yii::$app->user->isGuest) {
            return $this->goHome();
        }
```

```
    $model = new LoginForm();
    if ($model->load(Yii::$app->request->post()) && $model->login()) {
        return $this->goBack();
    }

    //$model->password = '';
    return $this->render('login', [
        'model' => $model,
    ]);
}

/**
 * Logout action.
 *
 * @return Response
 */
public function actionLogout()
{
    Yii::$app->user->logout();

    return $this->goHome();
}

public function actionSignup()
{
    $model = new SignupForm();

    if(Yii::$app->request->isPost)
    {
        $model->load(Yii::$app->request->post());
        if($model->signup())
        {
```

```
            return $this->redirect(['auth/login']) ;
        }
    }

    return $this->render('signup', ['model' => $model]);
}
public function actionTest()
{
    $user = User::findOne(1);
    Yii::$app->user->logout();

    if(Yii::$app->user->isGuest){
        echo "guest";
    }
    else{
        echo "no";
    }
}
}
```

Article.php:

```
<?php

namespace app\models;

use Yii;
use yii\data\Pagination;

/**
 * This is the model class for table "article".
 *
 * @property int $id
```

```
 * @property string|null $title
 * @property string|null $description
 * @property string|null $date
 * @property string|null $image
 * @property string|null $tag
 * @property int|null $viewed
 * @property int|null $category_id
 * @property int|null $user_id
 *
 * @property Category $category
 * @property Comment[] $comments
 * @property User $user
 */
class Article extends \yii\db\ActiveRecord
{
    /**
     * {@inheritdoc}
     */
    public static function tableName()
    {
        return 'article';
    }

    /**
     * {@inheritdoc}
     */
    public function rules()
    {
        return [
            [['title','description','tag','category_id','user_id'], 'required'],
            [['title','description'], 'string'],
            [['date'], 'date', 'format'=>'php:Y-m-d'],
            [['date'], 'default', 'value'=>date('Y-m-d')],
```

```
            [['viewed'], 'default', 'value'=>0],
            [['viewed', 'category_id', 'user_id'], 'integer'],
            [['user_id'], 'exist', 'skipOnError' => true, 'targetClass' => User::class, 'targetAttribute' => ['user_id' => 'id']],
            [['category_id'], 'exist', 'skipOnError' => true, 'targetClass' => Category::class, 'targetAttribute' => ['category_id' => 'id']],
        ];
    }

    /**
     * {@inheritdoc}
     */
    public function attributeLabels()
    {
        return [
            'id' => 'ID',
            'title' => 'Title',
            'description' => 'Description',
            'date' => 'Date',
            'image' => 'Image',
            'tag' => 'Tag',
            'viewed' => 'Viewed',
            'category_id' => 'Category ID',
            'user_id' => 'User ID',
        ];
    }

    /**
     * Gets query for [[Category]].
     *
     * @return \yii\db\ActiveQuery
     */
    public function getCategory()
```

```
{
    return $this->hasOne(Category::class, ['id' => 'category_id']);
}

/**
 * Gets query for [[Comments]].
 *
 * @return \yii\db\ActiveQuery
 */
public function getComments()
{
    return $this->hasMany(Comment::class, ['article_id' => 'id']);
}

/**
 * Gets query for [[User]].
 *
 * @return \yii\db\ActiveQuery
 */
public function getUser()
{
    return $this->hasOne(User::class, ['id' => 'user_id']);
}

public function saveImage($filename)
{
    $this->image = $filename;
    return $this->save(false);
}

public function deleteImage()
{
    $imageUploadModel = new ImageUpload();
```

```
    $imageUploadModel->deleteCurrentImage($this->image);
}

public function beforeDelete()
{
    $this->deleteImage();
    return parent::beforeDelete(); // TODO: Change the autogenerated stub
}

public function getImage()
{
    return ($this->image) ? '/uploads/' . $this->image : '/no-image.jpg';
}

public function getDate()
{
    return Yii::$app->formatter->asDate($this->date);
}

public static function getAll($pageSize = 5)
{
    // build a DB query to get all articles with status = 1
    $query = Article::find();

    // get the total number of articles (but do not fetch the article data yet)
    $count = $query->count();

    // create a pagination object with the total count
    $pagination = new Pagination(['totalCount' => $count, 'pageSize'=>$pageSize]);

    // limit the query using the pagination and retrieve the articles
    $articles = $query->offset($pagination->offset)
        ->limit($pagination->limit)
```

```
            ->all();

        $data['articles'] = $articles;
        $data['pagination'] = $pagination;

        return $data;
    }

    public static function getPopular()
    {
        return Article::find()->orderBy('viewed desc')->limit(3)->all();
    }

    public static function getRecent()
    {
        return Article::find()->orderBy('date desc')->limit(3)->all();
    }

    public function saveArticle()
    {
        $this->user_id = Yii::$app->user->id;
        $this->save();
    }

    public function viewedCounter()
    {
        $this->viewed += 1;
        return $this->save(false);
    }


}
```

Category.php:

```
<?php

namespace app\models;

use Yii;
use yii\data\Pagination;

/**
 * This is the model class for table "category".
 *
 * @property int $id
 * @property string|null $name
 *
 * @property Article[] $articles
 */
class Category extends \yii\db\ActiveRecord
{
    /**
     * {@inheritdoc}
     */
    public static function tableName()
    {
        return 'category';
    }

    /**
     * {@inheritdoc}
     */
    public function rules()
    {
```

```
    return [
        [['name'], 'required'],
        [['name'], 'string', 'max' => 255],
    ];
}

/**
 * {@inheritdoc}
 */
public function attributeLabels()
{
    return [
        'id' => 'ID',
        'name' => 'Name',
    ];
}

/**
 * Gets query for [[Articles]].
 *
 * @return \yii\db\ActiveQuery
 */
public function getArticles()
{
    return $this->hasMany(Article::class, ['category_id' => 'id']);
}

public function getArticlesCount()
{
    return $this->getArticles()->count();
}

public static function getAll()
```

```
    {
        return Category::find()->all();
    }

    public static function getArticlesByCategory($id)
    {
        // build a DB query to get all articles with status = 1
        $query = Article::find()->where(['category_id'=>$id]);

        // get the total number of articles (but do not fetch the article data yet)
        $count = $query->count();

        // create a pagination object with the total count
        $pagination = new Pagination(['totalCount' => $count, 'pageSize'=>6]);

        // limit the query using the pagination and retrieve the articles
        $articles = $query->offset($pagination->offset)
            ->limit($pagination->limit)
            ->all();

        $data['articles'] = $articles;
        $data['pagination'] = $pagination;

        return $data;
    }
}
```

CommentForm.php:

```
<?php

namespace app\models;
```

```
use Yii;
use yii\base\Model;

class CommentForm extends Model
{
    public $comment;

    public function rules()
    {
        return [
            [['comment'], 'required'],
            [['comment'], 'string', 'length' => [3,250]]
        ];
    }

    public function saveComment($article_id)
    {
        $comment = new Comment;
        $comment->text = $this->comment;
        $comment->user_id = Yii::$app->user->id;
        $comment->article_id = $article_id;

        $comment->date = date('Y-m-d');
        return $comment->save();
    }
}
```

Comment.php:

```
<?php
```

```
namespace app\models;

use Yii;

/**
 * This is the model class for table "comment".
 *
 * @property int $id
 * @property string|null $text
 * @property int|null $user_id
 * @property int|null $comment_id
 * @property int|null $article_id
 * @property string|null $date
 * @property int|null $delete
 *
 * @property Article $article
 * @property Comment $comment
 * @property Comment[] $comments
 * @property User $user
 */
class Comment extends \yii\db\ActiveRecord
{
    public function getDate()
    {
        return Yii::$app->formatter->asDate($this->date);
    }
    /**
     * {@inheritdoc}
     */
    public static function tableName()
    {
        return 'comment';
    }
```

```
    /**
     * {@inheritdoc}
     */
    public function rules()
    {
        return [
            [['user_id', 'comment_id', 'article_id', 'delete'], 'integer'],
            [['user_id', 'comment_id', 'text'], 'required'],
            [['date'], 'date', 'format'=>'php:Y-m-d'],
            [['date'], 'default', 'value'=>date('Y-m-d')],
            [['text'], 'string', 'max' => 255],
            [['article_id'], 'exist', 'skipOnError' => true, 'targetClass' => Article::class, 'targetAttribute' => ['article_id' => 'id']],
            [['user_id'], 'exist', 'skipOnError' => true, 'targetClass' => User::class, 'targetAttribute' => ['user_id' => 'id']],
            [['comment_id'], 'exist', 'skipOnError' => true, 'targetClass' => Comment::class, 'targetAttribute' => ['comment_id' => 'id']],
        ];
    }

    /**
     * {@inheritdoc}
     */
    public function attributeLabels()
    {
        return [
            'id' => 'ID',
            'text' => 'Text',
            'user_id' => 'User ID',
            'comment_id' => 'Comment ID',
            'article_id' => 'Article ID',
            'date' => 'Date',
            'delete' => 'Delete',
```

```
    ];
}

/**
 * Gets query for [[Article]].
 *
 * @return \yii\db\ActiveQuery
 */
public function getArticle()
{
    return $this->hasOne(Article::class, ['id' => 'article_id']);
}

/**
 * Gets query for [[Comment]].
 *
 * @return \yii\db\ActiveQuery
 */
public function getComment()
{
    return $this->hasOne(Comment::class, ['id' => 'comment_id']);
}

/**
 * Gets query for [[Comments]].
 *
 * @return \yii\db\ActiveQuery
 */
public function getComments()
{
    return $this->hasMany(Comment::class, ['comment_id' => 'id']);
}
```

```
    /**
     * Gets query for [[User]].
     *
     * @return \yii\db\ActiveQuery
     */
    public function getUser()
    {
        return $this->hasOne(User::class, ['id' => 'user_id']);
    }
}
```

singleArticle.php:

```
<?php

use yii\helpers\Url;
use app\controllers\CommentForm;
?>
<!--main content start-->
<div class="main-content">
    <div class="container">
        <div class="row">
            <div class="col-md-8">
                <article class="post">
                    <div class="post-thumb">
                        <a href="blog.html"><img src="<?= $article->getImage(); ?>" alt=""></a>
                    </div>
                    <div class="post-content">
                        <header class="entry-header text-center text-uppercase">
                            <h6><a href="<?= Url::toRoute(['site/category', 'id' => $article->category->id]) ?>"><?=
$article->category->name; ?></a></h6>

                            <h1 class="entry-title"><a href="<?= Url::toRoute(['site/view', 'id' => $article->id]) ?>"><?=
$article->title; ?></a></h1>
```

```
        </header>
        <div class="entry-content">
            <?= $article->description; ?>
        </div>

        <div class="social-share">
            <span class="social-share-title pull-left text-capitalize"><?= $article->getDate(); ?></span>
            <ul class="text-center pull-right">
                <li><a class="s-facebook" href="#"><i class="fa fa-facebook"></i></a></li>
                <li><a class="s-twitter" href="#"><i class="fa fa-twitter"></i></a></li>
                <li><a class="s-google-plus" href="#"><i class="fa fa-google-plus"></i></a></li>
                <li><a class="s-linkedin" href="#"><i class="fa fa-linkedin"></i></a></li>
                <li><a class="s-instagram" href="#"><i class="fa fa-instagram"></i></a></li>
            </ul>
        </div>
    </div>
</article>
<?php if (!empty($comments)) : ?>

    <?php foreach ($comments as $comment) : ?>
        <div class="bottom-comment"><!--bottom comment-->
            <div class="comment-img">
                <img class="img-circle" src="<?= $comment->user->image; ?>" alt="">
            </div>

            <div class="comment-text">
                <a href="#" class="replay btn pull-right"> Replay</a>
                <h5><?= $comment->user->name;?></h5>

                <p class="comment-date">
                    <?= $comment->getDate();?>
```

```
                    </p>


                    <p class="para"><?= $comment->text; ?></p>
                  </div>
                </div>
              <?php endforeach; ?>
            <?php endif; ?>

            <!-- end bottom comment-->


            <div class="leave-comment"><!--leave comment-->
              <h4>Leave a reply</h4>

<?php $form = \yii\widgets\ActiveForm::begin([
   'action'=>['site/comment', 'id'=>$article->id],
   'options'=>['class'=>'form-horizontal contact-form', 'role'=>'form']])?>
                  <div class="form-group">
                    <div class="col-md-12">
                      <?= $form->field($commentForm, 'comment')->textarea(['class'=>'form-control',
'placeholder'=>'Write Message'])->label(false)?>
                    </div>
                  </div>
                  <button type="submit" class="btn send-btn">Post Comment</a>
            <?php \yii\widgets\ActiveForm::end();?>
              </div><!--end leave comment-->
          </div>
          <?= $this->render('/partials/sidebar', [
            'popular' => $popular,
            'recent' => $recent,
            'categories' => $categories
          ]); ?>
```

```
        </div>
    </div>
</div>
<!-- end main content-->
```

Module.php:

```
<?php

namespace app\modules\admin;
use Yii;
use yii\filters\AccessControl;

/**
 * admin module definition class
 */
class Module extends \yii\base\Module
{
    /**
     * {@inheritdoc}
     */
    public $layout = '/admin';
    public $controllerNamespace = 'app\modules\admin\controllers';

    /**
     * {@inheritdoc}
     */
    public function init()
    {
        parent::init();

        // custom initialization code goes here
    }
    public function behaviors()
    {
        return [
            'access'    =>  [
                'class' =>  AccessControl::className(),
                'denyCallback'  =>  function($rule, $action)
                {
                    throw new \yii\web\NotFoundHttpException();
                },
                'rules' =>  [
                    [
                        'allow' =>  true,
                        'matchCallback' =>  function($rule, $action)
                        {
                            if (isset(Yii::$app->user->identity->login)){
                                return (Yii::$app->user->identity->login == 'stas@gmail.com');
                            }
                            else{
                                return false;
                            }
                        }
```

```
            ]
          ]
        ]
      ];
   }
}
```

Index.php:

```
<?php
use yii\widgets\LinkPager;
use yii\helpers\Url;
?>
<!--main content start-->
<div class="main-content">
   <div class="container">
      <div class="row">
         <div class="col-md-8">
            <?php foreach ($articles as $article):?>

               <article class="post">
                  <div class="post-thumb">
                     <a href="<?= Url::toRoute(['site/view', 'id'=>$article->id]); ?>"><img src="<?= $article-
>getImage(); ?>" alt=""></a>

                     <a href="<?= Url::toRoute(['site/view', 'id'=>$article->id]); ?>" class="post-thumb-overlay
text-center">
                        <div class="text-uppercase text-center">View Post</div>
                     </a>
                  </div>
                  <div class="post-content">
                     <header class="entry-header text-center text-uppercase">
                        <h6><a href="<?=Url::toRoute(['site/category', 'id'=>$article->category->id]) ?>"><?=
$article->category->name; ?></a></h6>

                        <h1 class="entry-title"><a href="<?= Url::toRoute(['site/view', 'id'=>$article->id]); ?>"><?=
$article->title; ?></a></h1>


                     </header>
                     <div class="entry-content">
                        <p> <?=$article->description; ?>
                        </p>

                        <div class="btn-continue-reading text-center text-uppercase">
                           <a href="<?= Url::toRoute(['site/view', 'id'=>$article->id]); ?>" class="more-
link">Continue Reading</a>
                        </div>
                     </div>
                     <div class="social-share">
                        <span class="social-share-title pull-left text-capitalize"><a href="#"></a><?= $article-
>getDate(); ?></span>
                        <ul class="text-center pull-right">
                           <li><a class="s-facebook" href="#"><i class="fa fa-eye"></i></a></li> <?= $article-
>viewed; ?>
                        </ul>
                     </div>
```

```
                </div>
            </article>

        <?php endforeach; ?>


        <?php
        echo LinkPager::widget([
            'pagination' => $pagination,
        ]);
        ?>
    </div>
    <?=$this->render('/partials/sidebar', [
        'popular'=>$popular,
        'recent'=>$recent,
        'categories'=>$categories
    ]); ?>        </div>
</div>
</div>
<!-- end main content-->
<!--footer start-->
```

Layounts/admin.php:

```
<?php

/** @var yii\web\View $this */
/** @var string $content */

use app\assets\AppAsset;
use app\widgets\Alert;
use yii\bootstrap5\Breadcrumbs;
use yii\bootstrap5\Html;
use yii\bootstrap5\Nav;
use yii\bootstrap5\NavBar;

AppAsset::register($this);

$this->registerCsrfMetaTags();
$this->registerMetaTag(['charset' => Yii::$app->charset], 'charset');
$this->registerMetaTag(['name' => 'viewport', 'content' => 'width=device-width, initial-scale=1, shrink-to-
fit=no']);
$this->registerMetaTag(['name' => 'description', 'content' => $this->params['meta_description'] ?? '']);
$this->registerMetaTag(['name' => 'keywords', 'content' => $this->params['meta_keywords'] ?? '']);
$this->registerLinkTag(['rel' => 'icon', 'type' => 'image/x-icon', 'href' => Yii::getAlias('@web/favicon.ico')]);
?>
<?php $this->beginPage() ?>
<!DOCTYPE html>
<html lang="<?= Yii::$app->language ?>" class="h-100">
<head>
    <title><?= Html::encode($this->title) ?></title>
    <?php $this->head() ?>
</head>
<body class="d-flex flex-column h-100">
<?php $this->beginBody() ?>

<header id="header">
```

```
    <?php
    NavBar::begin([
        'brandLabel' => 'blog-pets',
        'brandUrl' => Yii::$app->homeUrl,
        'options' => ['class' => 'navbar-expand-md navbar-dark bg-dark fixed-top']
    ]);
    echo Nav::widget([
        'options' => ['class' => 'navbar-nav'],
        'items' => [
            ['label' => 'Users', 'url' => ['/admin/user/index']],
            ['label' => 'Categories', 'url' => ['/admin/category/index']],
            ['label' => 'Articles', 'url' => ['/admin/article/index']],
            ['label' => 'Comments', 'url' => ['/admin/comment/index']],
        ]
    ]);
    NavBar::end();
    ?>
</header>

<main id="main" class="flex-shrink-0" role="main">
    <div class="container">
        <?php if (!empty($this->params['breadcrumbs'])): ?>
            <?= Breadcrumbs::widget(['links' => $this->params['breadcrumbs']]) ?>
        <?php endif ?>
        <?= Alert::widget() ?>
        <?= $content ?>
    </div>
</main>

<footer id="footer" class="mt-auto py-3 bg-light">
    <div class="container">
        <div class="row text-muted">
            <div class="col-md-6 text-center text-md-start">&copy; My Company <?= date('Y') ?></div>
            <div class="col-md-6 text-center text-md-end"><?= Yii::powered() ?></div>
        </div>
    </div>
</footer>

<?php $this->endBody() ?>
</body>
</html>
<?php $this->endPage() ?>
```

Layount/main.php:

```
<?php

/** @var yii\web\View $this */
/** @var string $content */

use app\assets\AppAsset;
use app\assets\PublicAsset;
use app\widgets\Alert;
use yii\bootstrap5\Breadcrumbs;
use yii\bootstrap5\Html;
use yii\bootstrap5\Nav;
use yii\bootstrap5\NavBar;
```

```
use yii\helpers\Url;

PublicAsset::register($this);

$this->registerCsrfMetaTags();
$this->registerMetaTag(['charset' => Yii::$app->charset], 'charset');
$this->registerMetaTag(['name' => 'viewport', 'content' => 'width=device-width, initial-scale=1, shrink-to-
fit=no']);
$this->registerMetaTag(['name' => 'description', 'content' => $this->params['meta_description'] ?? '']);
$this->registerMetaTag(['name' => 'keywords', 'content' => $this->params['meta_keywords'] ?? '']);
$this->registerLinkTag(['rel' => 'icon', 'type' => 'image/x-icon', 'href' => Yii::getAlias('@web/favicon.ico')]);
?>
<?php $this->beginPage() ?>
<!DOCTYPE html>
<html lang="<?= Yii::$app->language ?>" class="h-100">
<head>
    <title><?= Html::encode($this->title) ?></title>
    <?php $this->head() ?>
</head>
<body class="d-flex flex-column h-100">
<?php $this->beginBody() ?>

<nav class="navbar main-menu navbar-default">
    <div class="container">
        <div class="menu-content">
            <!-- Brand and toggle get grouped for better mobile display -->
            <div class="navbar-header">
                <button type="button" class="navbar-toggle collapsed" data-toggle="collapse"
                        data-target="#bs-example-navbar-collapse-1">
                    <span class="sr-only">Toggle navigation</span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <a class="navbar-brand" href="/"><img src="/public/images/pawprint.png" alt=""></a>
            </div>


            <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">

                <ul class="nav navbar-nav text-uppercase">
                    <li><a data-toggle="dropdown" class="dropdown-toggle" href="/">Home</a>

                    </li>
                </ul>
                <div class="i_con">
                    <ul class="nav navbar-nav text-uppercase">
<!--                        <li><a href="/site/login">Login</a></li>-->
<!--                        <li><a href="/site/signup">Register</a></li>-->
                        <?php if(Yii::$app->user->isGuest):?>
                            <li><a href="<?= Url::toRoute(['auth/login'])?>">Login</a></li>
                            <li><a href="<?= Url::toRoute(['auth/signup'])?>">Register</a></li>
                        <?php else: ?>
                            <?= Html::beginForm(['/auth/logout'], 'post')
                            . Html::submitButton(
                                'Logout (' . Yii::$app->user->identity->name . ')',
                                ['class' => 'btn btn-link logout', 'style'=>"padding-top:10px;"]
                            )
                            . Html::endForm() ?>
                        <?php endif;?>
```

```
                </ul>
            </div>

        </div>
        <!-- /.navbar-collapse -->
      </div>
   </div>
   <!-- /.container-fluid -->
</nav>


<?=$content?>

<footer class="footer-widget-section">
   <div class="footer-copy">
      <div class="container">
         <div class="row">
            <div class="col-md-12">
               <div class="text-center">&copy; 2023 <a href="#">PETS-BLOG, </a> by <a href="#"> </a>
               </div>
            </div>
         </div>
      </div>
   </div>
</footer>

<?php $this->endBody() ?>
</body>
</html>
<?php $this->endPage() ?>
```

Sidebar.php:

```
<?php
use yii\helpers\Url;
?>
<div class="col-md-4" data-sticky_column>
   <div class="primary-sidebar">

      <aside class="widget">
         <h3 class="widget-title text-uppercase text-center">Popular Posts</h3>

         <?php foreach ($popular as $article):?>

            <div class="popular-post">
               <a href="<?=Url::toRoute(['site/view', 'id'=>$article->id]) ?>" class="popular-img"><img src="<?=
$article->getImage(); ?>" alt="">

                  <div class="p-overlay"></div>
               </a>

               <div class="p-content">
                  <a href="<?=Url::toRoute(['site/view', 'id'=>$article->id]) ?>" class="text-uppercase"><?=
$article->title; ?></a>
                  <span class="p-date"><?= $article->getDate(); ?></span>

               </div>
```

```
            </div>


          <?php endforeach;?>

        </aside>
        <aside class="widget pos-padding">
          <h3 class="widget-title text-uppercase text-center">Recent Posts</h3>

          <?php foreach ($recent as $article): ?>

            <div class="thumb-latest-posts">


              <div class="media">
                <div class="media-left">
                  <a href="<?=Url::toRoute(['site/view', 'id'=>$article->id]) ?>" class="popular-img"><img
src="<?= $article->getImage(); ?>" alt="">
                    <div class="p-overlay"></div>
                  </a>
                </div>
                <div class="p-content">
                  <a href="<?=Url::toRoute(['site/view', 'id'=>$article->id]) ?>" class="text-uppercase"><?=
$article->title; ?></a>
                  <span class="p-date"><?= $article->getDate(); ?></span>
                </div>
              </div>
            </div>


          <?php endforeach; ?>


        </aside>
        <aside class="widget border pos-padding">
          <h3 class="widget-title text-uppercase text-center">Categories</h3>
          <ul>
            <?php foreach ($categories as $category): ?>
              <li>
                <a href="<?=Url::toRoute(['site/category', 'id'=>$category->id]) ?>"> <?=$category->name; ?>
</a>
                <span class="post-count pull-right"> <?=$category->getArticlesCount(); ?> </span>
              </li>
            <?php endforeach; ?>

          </ul>
        </aside>
      </div>
    </div>
```