МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Зав. кафедри

Давиденко Є. О. _____

« __ » _____ 2022 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

«ІНФОРМАЦІЙНИЙ ВЕБЗАСТОСУНОК ІНТЕРНЕТ-ВИДАННЯ»

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 409. 21810907

Студент _____Н. Ю. Воронцова
підпис
«__» _____ 2022 р.

Керівник д-р. техн. наук, доцент _____А. В. Швед
підпис
«__» _____ 2022 р.

Консультант канд. техн. наук, доцент (б. в. з.) _____А. О. Алексєєва
підпис
«__» _____ 2022 р.

Миколаїв – 2022

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних;

ЗМІ – засоби масової інформації;

ПЗ – програмне забезпечення;

ПКМ – права кнопка миші;

СУБД – система управління базами даних.

HEX – (from the Greek) six and the fractions (from the Latin) decimal – the tenth. Hexadecimal;

RGB –red, green, blue;

CSS – Cascading Style Sheets;

FK – foreign key;

Id – identity document;

HTML – HyperText Markup Language;

HTTP – Hypertext Transfer Protocol;

UML – Unified Modeling Language;

DTO – data transfer object;

MVC – Model-view-controller;

API – Application Programming Interface.

ВСТУП

Актуальність:

Якщо раніше журнал виходив у друкованому вигляді, то зараз він базується на інтернет-сайті з періодичним оновленням усіх матеріалів. І якщо раніше це була лише копія в Інтернеті, то тепер це повноцінна структура, яка включає блоги, форуми, періодичні видання, платну та безкоштовну частину для перегляду та багато іншого, що навіть не перетинається з паперовим виданням.

Відстань між ЗМІ та читачами значно скоротилась завдяки інтернет – журналістиці. Читачі отримали змогу не тільки читати новини, але й оцінювати, коментувати, підмічати помилки, висувати нові ідеї. Стає дедалі очевидно, що все більше впливає на популярність та конкурентоспроможність медіа активна взаємодія з читачами.

Освічена аудиторія, незалежно-мислячі журналісти, поширення інформації без просторових та цензурних обмежень і великий попит на таку інформацію забезпечують успішний розвиток інтернет-ЗМІ. На сьогоднішній день вебжурналістика прогресує найбільш динамічно і тому інтернет-ЗМІ стали більш адаптованими до сучасного інформаційного світу. Враховуючи велику кількість користувачів в інтернеті, можна сказати, що інтернет-видання гарні великі можливості для успішного розвитку у національному інформаційному просторі.

Практичне значення:

Основною ідеєю новинного порталу є формування критичного і розширеного обсягу інформаційних послуг з метою залучення якомога більшої аудиторії. Таким чином, новинний портал для користувача – це онлайн-медіа-ресурс, який надає інтерактивні новинні сервіси, які працюють у межах єдиного ресурсу, служать точками доступу до новинної інформації та допомагають шукати її в Інтернеті.

Метою роботи є підвищення рівня інформаційної обізнаності суспільства, формування критичного ставлення медіа-аудиторії до отримуваної інформації шляхом розширення спектру інформаційно-аналітичних послуг на ринку українських інтернет-видань.

Об'єктом роботи є інформаційні технології та інструментальні засоби розробки програмного забезпечення інтернет-видання..

Предметом роботи є змістовно-концептуальні особливості подання журналістських онлайн-матеріалів при організації роботи інформаційного інтернет-видання.

Завдання роботи:

Для досягнення мети треба вирішити наступні завдання:

1. Визначення структурної схеми сайту - розташування розділів, контенту і навігації;
2. Вебдизайн - створення графічних елементів макету сайту, стилів і елементів навігації;
3. Розробка програмного коду, модулів, бази даних і інших елементів сайту необхідних в проекті:

Система повинна працювати швидко і без перебоїв.

Повинно бути створено:

1) Зручна панель (сторінка) адміністратора,
2) головна сторінка з слайдером останніх новин та з переліком заголовків статтей та картинкою,
3) сторінка реєстрації та логіну,
4) пошук на сайті,
5) реалізація функціоналу вподобання та коментування статтей,
6) сторінка для виведення статті з різними режимами перегляду, який користувач зможе вибрати для додаткової зручності.

4. Тестування застосунку.

1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ ІНТЕРНЕТ-ЖУРНАЛІСТИКИ В СУЧАСНОМУ МЕДІА-ПРОСТОРІ

Новини завжди будуть залучати і приваблювали аудиторію. Саме тому такі сайти є одними з більш відвідуваних. Через те велика кількість власників сайтів прагнуть до такого формату подачі інформації до читачів. При цьому неважливим фактором є місцеві це новини або факти світового формату.

Інтернет є ефективним засобом комунікації, що допомагає обмінюватися інформацією між журналістами та наладжувати зворотний зв'язок із читачами, використовуючи при цьому різноманітні засоби комунікації, наприклад електронну пошту, форуми, коментарі, чати, сторінки у соціальних мережах, систему оцінок, інтерактивні опитування та інші [4].

Оперативність є однією з головних переваг мережі. Інтернет-видання значніше обганяють за швидкістю передачі інформації телебачення, пресу, та радіо. Контент можна швидко додати, оновити, змінити чи видалити, навіть через кілька хвилин після події або в один і той же час. Також одною з важливих переваг є зручність інтернет-середовища. Це зумовлюється тим, що кожен користувач має право самостійно обирати час користування послугами мережі чи обирати способи отримання та формат інформації.

1.1. Опис предметної сфери інтернет-медіа

Інтернет-ЗМІ – це сайти з постійним оновленням інформації, які відвідують відносно велика кількість людей. Вони надають своєму читачеві або глядачеві саме ту інформацію, яку вважають соціально значущою. Велика кількість інтернет-ЗМІ оновлюється щогодини або за необхідністю, тому кількість новин доходить до кількох сотень. Через високу оперативність, інтернет-ЗМІ нерідко використовуються як джерела інформації для звичайних ЗМІ.

Інтернет-журналістика – це різновидність журналістики, що визначається розповсюджуванням журналістських матеріалів через мережу Інтернет.

Порівняно з традиційними Інтернет – ЗМІ мають велику кількість переваг. Для появи статті в мережі необхідно натиснути кілька кнопок і вона миттєво з'являється в інтернеті. Окрім цього, будь-який читач має змогу залишити свій коментар – думку відносно того чи іншого журналістського матеріалу. Також влаштовуються дискусії, які присвячені найактуальнішим темам, а газети позбавлені такої можливості. Інтернет-видання – це практичність і зручність. Набагато простіше прочитати новину, скориставшись комп'ютером або мобільним телефоном, ніж шукати газети у найближчих кіосках та читати новини не сьогоднішнього дня. Також немало важливі плюси – це гіпертекстуальність, електронний архів, швидкий пошук необхідної інформації, персоналізація.

Аудиторії інтернет-видань необхідно мати схожість з такими критеріями маси: складатися з колосальної кількості розосереджених у просторі індивідів; бути відкритою, ситуативною, непостійною, анонімною, неоднорідною за соціальними та демографічними покажчиками. Інтернет надає нам можливість для втілювання двосторонніх зв'язків – журналіста з аудиторією, на відміну від радіо, преси та телебачення, які спілкуються з читачами у режимі монологу.

Інтернет-видання мають велику перевагу серед інших ЗМІ, а саме надання аудиторії альтернативи у виборі анонімності чи персоналізації. Користувач може зареєструватися на сайті під нік-неймом, не називаючи свого реального прізвища. Анонімність надає кожному користувачу інтернет-аудиторії значно більше свободи у висловленні своїх думок, ніж у випадку обов'язкової ідентифікації особистості. На сьогоднішній день ще все таки найбільша кількість аудиторії залишається на телевізійних каналах, але кількісні показники інтернет-аудиторії випередили друковані ЗМІ та радіо та постійно збільшуються на користь інтернет-видань.

1.2. Типологія жанрів інтернет-журналістики. Типи та види сучасних інтернет-видань

Важливим аспектом інтернет-видань є спосіб подання тої чи іншої інформації, бо інформаційна інтернет-конкуренція активно бореться за увагу аудиторії. З огляду на різноманітність інтернет-користувачів, головним залишається зміст повідомлень та його подання через постійну взаємодію з цільовою аудиторією, дослідження її як об'єкта та рівноправного учасника діалогу [2].

На сьогоднішній день існує кілька варіантів класифікації інтернет-видань у медійному інтернет-просторі за різними характеристиками. Дослідники виділяють три основні групи інтернет-ЗМІ: паперово-мережеві – копії традиційних ЗМІ; мережево-паперові – модифіковані онлайн-версії традиційних ЗМІ, які не копіюють їх, а здійснюють особисту інформаційну політику; суто-мережеві – проекти, що виникли онлайн і не мають прототипів.

Можна виділити такі варіанти класифікації інтернет-видань (рис. 1.1).

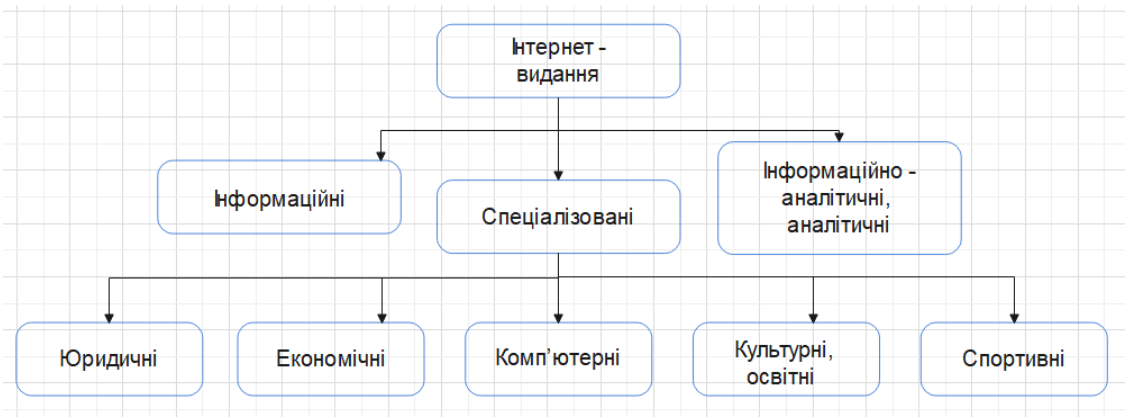


Рисунок 1.1 – Класифікація інтернет-видань

Розділяють інтернет-ЗМІ по таким типологічним ознакам:

1. за ступенем професійності – професійні й аматорські,
2. за доступністю для читачів – загальнодоступні та з обмеженим доступом,
3. за тематикою – монотематичні та політематичні,

4. за належністю – приватні (проекти медійних груп, бізнес-компаній чи політичних партій), державні та незалежні,

5. за аудиторією – загальні та спеціалізовані,

6. за оновлювальністю – регулярні, нерегулярні, по мірі надходження інформації.

Як правило, інтернет-ЗМІ електронних або друкованих українських видань особливо нічим не відрізняються від власного інтернет-видання. Вони подають однакову інформацію, тому її можна прочитати у газеті, побачити на телебаченні чи почути по радіо. Проте інтернет-версії дають змогу друкованим та електронним виданням динамічно співпрацювати з аудиторією. А саме створювати форуми сайтів, архіви, матеріали які можна побачити, почути чи прочитати у будь-який час та у будь-якому місці.

1.3. Аналіз та порівняльна характеристика сучасних інтернет-видань

Для аналізу сучасних інтернет-видань візьмемо три популярні вебсайти, а саме українські вебпортали «Оглядач», «НикВести» та «ukr.net».

Аналог №1

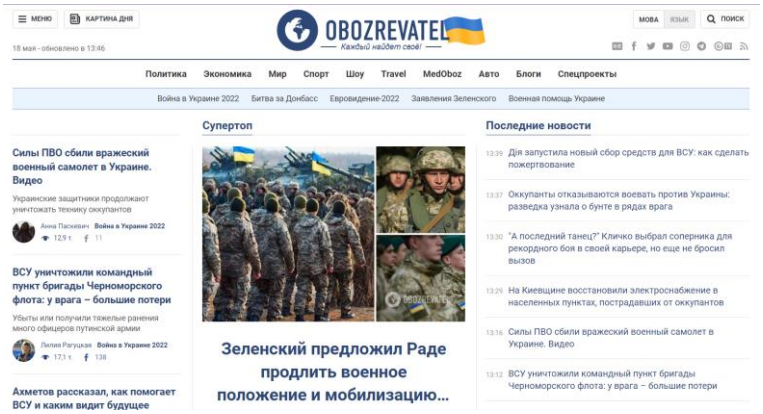


Рисунок 1.2 – Головна сторінка вебпорталу

1) **Назва**: Оглядач.

2) **Виробник**: український вебпортал.

3) **Архітектура**: клієнт-сервер.

4) **Перелік функцій, характеристик**:

— Пошук по новинам та публікаціям;
— вибір мови;
— тематичні модулі;
— топ новин;
— головна сторінка та сторінка новин за «сьогодні»;
— перехід на інші соціальні мережі;
— є своя телепрограма та радіо;
— можливість реагувати на статті та коментарі інших читачів;
— сортування коментарів.

5) **Переваги**:

— Швидкий пошук;
— багато новин із різних сфер життя;
— можливість поставити оцінку та залишити коментарі;
— можливість перейти на інші соціальні мережі;
— розподіл по топ-темам;
— сайт складається з тематичних модулів.

6) **Недоліки**:

— Незручна навігація;
— занадто багато інформації на головній сторінці.

Аналог №2

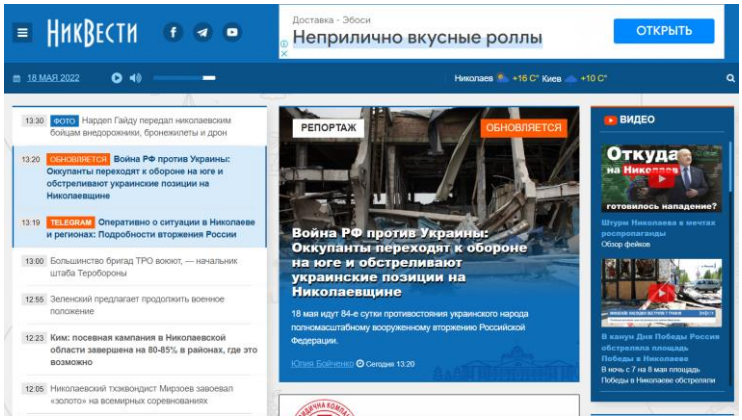


Рисунок 1.3 – Головна сторінка вебпорталу

1) **Назва**: НикВести.
2) **Виробник**: український вебпортал (м.Миколаїв).
3) **Архітектура**: клієнт-сервер.
4) **Перелік функцій, характеристик**:
   - Пошук по новинам та зручна навігація;
   - мова – російська;
   - тематичні модулі;
   - гарний та зрозумілий інтерфейс;
   - фоторепортажі;
   - можливість перегляду старих новин завдяки архіву;
   - можливість перегляду блогів людей Миколаєва.
5) **Переваги**:
   - Швидкий пошук;
   - блоги;
   - комікси;
   - розподіл по топ-темам.
6) **Недоліки**:
   - Немає можливості спілкування між читачами, коментування та реагування на новини.

Аналог №3

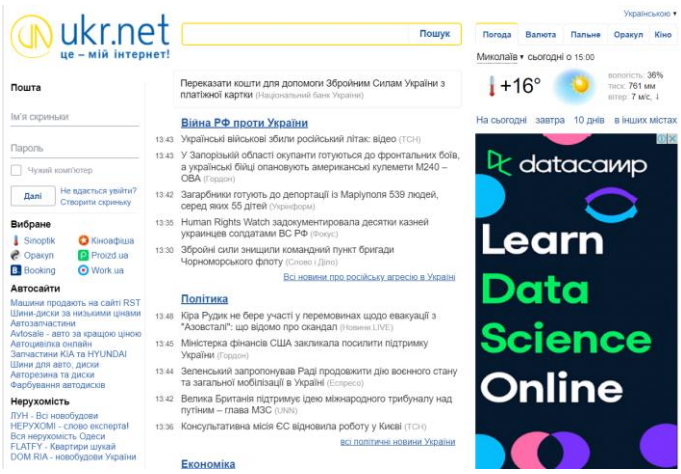


Рисунок 1.4 – Головна сторінка вебпорталу

1) **Назва**: ukr.net.
2) **Виробник**: український вебпортал, сервіс електронної пошти.
3) **Архітектура**: клієнт-сервер.
4) **Перелік функцій, характеристик**:
   - Пошук по новинам;
   - вибір мови;
   - тематичні модулі;
   - збір новин з інших новинних порталів;
   - є сервіс електронної пошти;
   - можливість перегляду новин;
   - можливість перегляду курсу валюти та ціни на пальне;
   - гороскоп;
   - можливість перегляду новинок кіно.
5) **Переваги**:
   - Швидкий пошук;
   - банер по місцю положення - погода;
   - швидкий доступ до курсу валюти та цін на пальне;
   - розподіл по топ-темам;
   - використання пошти.
6) **Недоліки**:
   - Занадто багато тексту та іншої інформації на сторінці.

Отже, важливою рисою розробки інтернет-видання є зручний для користувачів інтерфейс, читабельність новин та можливість взаємодіяти з читачами. Для зручного використання сайту треба більш зрозуміло розміщувати статті, помістити поле для коментарів читачів та не загромаджувати головну сторінку іншою інформацією.

1.4. Специфікація вимог до програмного забезпечення вебзастосунку інтернет-видання

Призначення ПЗ:

Новини завжди приваблювали людей. І тому, на теперішній час, такі проекти є одними з найбільш відвідуваних. Багато власників сайтів прагнуть саме до такого формату подачі інформації користувачам інтернету. І зовсім неважливо – місцеві це новини чи світового формату.

ПЗ розроблено з метою подальшого його використання у ознайомленні з останніми подіями, новинами.

Ролі користувачів системи:

1) Адміністратор – це користувач програмного забезпечення, який виконує налаштування та управління функціоналом. Він займається додаванням, редагуванням і видаленням статей.

2) Користувач (читач) – перегляд статтей, пошук новин на сайті, реагування та коментування статтей при умові реєстрації.

Система повинна мати реалізацію таких функцій:

1) Відслідковування кількості входів.

2) Можливість реєстрування та логіну.

3) Надання читачам можливості коментувати статті (при умові реєстрації).

4) Надання читачам реагувати на статті (при умові реєстрації).

5) Додавання, видалення та зміна новин з панелі адміністратора.

6) Банер останніх 3 новин за день на головній сторінці.

7) Пошук по сайту.

8) Можливість вибору дизайну відображення статті.

Було створено специфікації декількох функціональних вимог, які описані в таблицях 1.1 – 1.5.

Таблиця 1.1 – Опис функціональної вимоги

Назва	Реєстрація клієнту
Опис	Після реєстрації клієнту, його дані зберігаються в базі даних.

Таблиця 1.2 – Опис функціональної вимоги

Назва	Логін клієнту
Опис	Після логіна клієнта на сайті можливість коментувати та ставити лайки під статтею.

Таблиця 1.3 – Опис функціональної вимоги

Назва	Пошук на сайті
Опис	Пошук та відображення потрібної інформації на сторінці.

Таблиця 1.4 – Опис функціональної вимоги

Назва	Додавання статті
Опис	Після створення нової статті через адмін-панель, дані статті зберігаються і відображаються на сайті.

Таблиця 1.5 – Опис функціональної вимоги

Назва	Вподобати статтю
Опис	Після натискання на іконку серця біля статті дані зберігаються і відображається на сайті збільшена кількість вподобань.

Також були виявлені деякі нефункціональні вимоги, які описані в таблицях 1.6 – 1.9.

Таблиця 1.6 – Опис нефункціональної вимоги

Назва	Підтримка середовища .NET Core
Опис	Має підтримувати кросплатформене середовище . NET Core.

Таблиця 1.7 – Опис нефункціональної вимоги

Назва	Швидкодія
Опис	Запити повинні мати малий час відповіді.

Таблиця 1.8 – Опис нефункціональної вимоги

Назва	Адаптивність
Опис	Застосунок має правильно відображати інформацію на різних пристроях.

Таблиця 1.9 – Опис нефункціональної вимоги

Назва	Потужність
Опис	Можливість мати велику кількість одночасних активних користувачів.

Висновки до розділу 1

Інтернет-ЗМІ виявилися одним із найважливіших явищ у новітньому медіа-просторі та світовій культурі. Інтернет-журналістика набирає обертів дуже швидко з моменту своєї появи і до сьогодні. На даний момент інтернет-видання займають вагоме місце у медіа-просторі, яке розриває уявлення традиційної журналістики.

У Розділі 1 представлено аналіз предметної сфери інтернет-журналістики в сучасному медіа-просторі, описано типологію жанрів інтернет-журналістики, її типи та види, сформовано об’єкт та предмет розроблювальної системи, виділені основні задачі. Було зроблено аналіз популярних вебпорталів, які мають схожу направленість.

Сформовані основні функціональні вимоги (перелік функцій або сервісів, які повинна надавати система, а також обмежень на дані і поводження системи при їхньому виконанні) та нефункціональні вимоги (визначають умови виконання функцій у середовищі, яке безпосередньо не пов'язане з функціями, а відбивають потреби користувачів щодо їх виконання).

2 ОСОБЛИВОСТІ ПІДГОТОВКИ КОНТЕНТУ ІНТЕРНЕТ-ВИДАНЬ

Контент можна поділити на: текстовий і мультимедійний. І кожен з них по-різному впливає на просування сайту та має свої переваги.

Текстовий контент можна розділити на такі дві групи: статистичний і динамічний.

1. Статистичний контент – це навчальні, продуктові, інформаційні сторінки. А саме, це текст, який розміщується на невизначений проміжок часу.

2. Динамічний – це сайти, які мають завдання захопити увагу відвідувачів сайту, результатом таких сайтів є активність користувачів у вигляді коментарів та відгуків.

Мультимедійний контент — це все те, що знаходиться на сайті, окрім тексту. Хоча картинки та відео індексуються пошуком гірше, даний вид контенту може сильно вплинути на поведінку відвідувачів сайту.

2.1 Вебкольори. Вплив кольору на сприймання інформації

Перше враження – головний момент при сприйнятті сайту користувачем. Воно стає відправною точкою для рішення, залишитися на сайті чи шукати інформацію на іншому сайті. Сайт, який має важку атмосферу, відштовхує користувача, і як наслідок — у користувача з’являється бажання залишити сайт, що призводить до втрати потенційного споживача [3].

Кольори у вебдизайні викликають у користувачів певні відчуття. Кожен колір прив'язаний до емоцій людини. Найкращі кольори для вебсторінок – червоний, помаранчевий, жовтий, зелений, синій та фіолетовий. Їх можна поєднувати між собою, але не більше трьох одночасно. Найкращим варіантом є розведення акцентних кольорів нейтральними (білий, чорний, сірий) [5].

Рекомендації для дизайну:

– 60% площі сторінок має припадати на основний відтінок;

– 30% відводять для вторинних. Під вторинними мають на увазі акцентні відтінки.

В розробці вебзастосунку буде використовуватися колірна модель HEX. HEX – (від грецького) шість, і частки (від латинського) -decimal – десятий, тобто шістнадцятковий. Це кодування базується на моделі RGB. У моделі RGB кольори отримують змішуванням трьох основних кольорів — червоного, зеленого і синього. Застосовується у більшості пристроїв, що мають кольорові екрани — моніторах, телевізорах, планшетах, смартфонах та навіть розумних годинниках. [6].

Під час розробки дизайну було вирішено використовувати такі кольори, як синій – #286CC7 (рис. 2.1), жовтий – #f0ad4e (рис. 2.2) та нейтральний – білий колір.

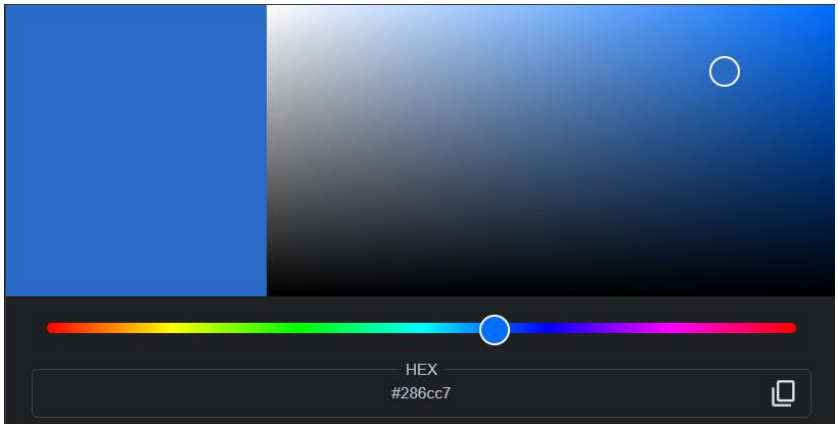


Рисунок 2.1 – Синій колір для застосунку

Синій колір – допоможе підвищити лояльність, завоювати довіру. Колір заспокоює, надає впевненості, почуття надійності.

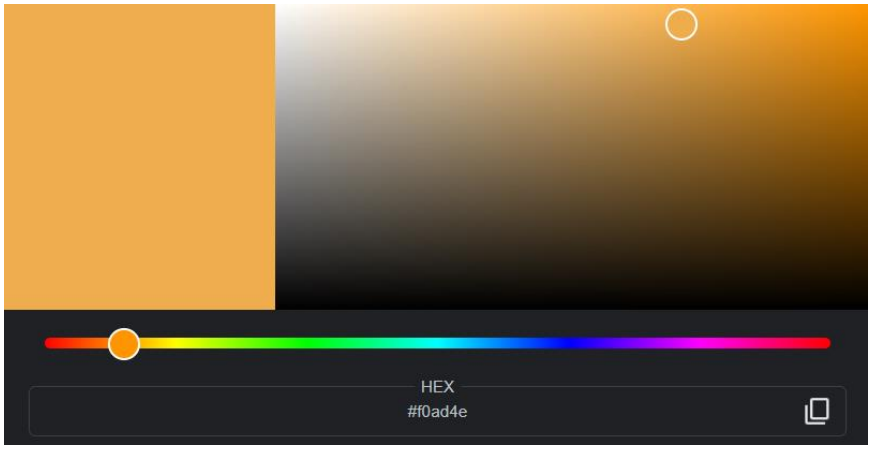


Рисунок 2.2 – Жовтий колір для застосунку

Жовтий – відтінок щастя, енергії. Може асоціюватись із статусом, солідністю, авторитетом. Білий дасть змогу сконцентруватися на прочитанні матеріалу.

2.2 Адаптація медіаконтенту до вебсередовища сайтів

Особливості вебсередовища сайтів впливають на зміст і зовнішній вид медіаконтенту. Як засіб у масовій комунікації інтернет-ЗМІ формує нові вимоги до створення медіатекстів.

Текст має бути інформативним і при цьому залишатися лаконічним і ємним за змістом. При адаптації матеріалу необхідно знижувати обсяг тексту на 50% порівняно з матеріалом у друкованому виданні або ж якщо це самостійне видання зміст повинен бути викладений коротко та ясно.

1. Інформативний заголовок.

Традиційні газетні заголовки для інтернет-видань зовсім не підходять. У друкованому виданні людина бачить всю новину цілком. В інтернеті є тільки заголовок і всього декілька секунд на його прочитання. Для інтернет-ресурсів характерно багаторівневе подання інформації та гіпертекстуальність, з чого випливає, що тексти. заголовків на сайтах повинні бути інформативними та повинні максимально розкривати тему, оскільки винесені на верхній структурний рівень читання матеріалу. У протилежному випадку пошук читачем потрібної інформації буде значно трудніший [7].

2. Використання ілюстративного матеріалу.

Користувачі інтернет-ресурсів «люблять» очима, тому зараз важко знайти матеріал на сайті, котрий би не доповнювався картинками, фото та іншим ілюстративним матеріалом. На практиці, читач насамперед звертає увагу на картинку. Тому треба ретельно підбирати відповідний ілюстративний матеріал: він має привертати увагу, бути оригінальним та повинен відображати тематику.

3. Розташування тексту на зображенні.

Щоб зацікавити читачів та зробити інформацію більш сприйнятною рекомендується розташовувати заголовки на зображеннях, що стосуються

тематики матеріалу. Цей прийом дозволяє зробити текст більш ефектним і помітним для очей читачів.

4. Доповнення тексту відео та аудіоматеріалом.

Статті, доповнені відео чи аудіоматеріалом, привертають більше уваги читачів і додають різноманітності у контент, що публікується.

5. Структурування інформації та розстановка акцентів.

Для більш легкого та швидкого сприйняття інформації зазвичай використовують заголовки, підзаголовки, перерахування. Один абзац повинен містити у собі одну завершену думку. На сайті, за допомогою CSS можна оформлювати текстову інформацію, використовуючи різні шрифти, рамки, фон, виділяти кольором та багато іншого. Також є можливість форматувати текст, наприклад використовувати прогалини між абзацами, писати слова великими літерами, підсвічувати важливе, а також вставляти різні смайли, щоб виразити емоції.

6. Спрощення синтаксичних конструкцій.

З метою більш легкого сприйняття сенсу переданої інформації необхідно спрощувати синтаксичні конструкції речень. Громіздкі речення не дають читачу швидко сприймати інформацію та знижують концентрацію уваги, що може стати причиною втрати відвідувача з вебсторінки.

7. Використання композиційного принципу «перевернута піраміда».

Суть цієї моделі залучати увагу користувача за допомогою акцентування на головному ще на самому початку статті, бо це є головною метою – «захопити» користувача та відразу представити йому саму важливу інформацію та заощадити його час [7].

2.3 Методи захисту контенту в інтернеті

Відповідно до Статті 50 Закону України «Про захист авторських та суміжних прав» здійснюється захист внутрішнього і зовнішнього наповнення веб-сайту. Однак застосування цього нормативного акту не завжди є доцільним

в сучасності, адже поняття, такі як – вебсайт, контент та інші, не є визначеними [8].

Спершу варто з’ясувати, який вид контенту потрібно захистити – мультимедійний чи текстовий. Найбільш дієвими способами є застосування «водяних знаків» (рис. 2.3) на авторські зображення, заборона виділення тексту на або клацання правою кнопкою миші [8]. Таким чином можна захистити як і текст, так і мультимедію.

Рисунок 2.3 – Приклад «водяного знаку»

Багато власників сайтів не замислюються над тим, щоб найняти копірайтерів і заплатити за написання текстів. В таких випадках треба замислюватися над захистом контенту від копіювання.

Контент – основа всього вебсайту, від якого залежать результати просування сайту. Найчастіший випадок: текст на вашому блозі не встиг проіндексуватися, а популярний майданчик швидко копіює і збирає всі перегляди. Тому, пошукові системи вважають першоджерелом сайт популярного конкурента, а не ваш. Так виходить за рахунок миттєвої індексації. Звісно можна зв'язатися з власником сайту та попрохати видалити текст або вставити гіперпосилання на першоджерело, але, не факт, що власник погодиться.

Є деякі способи, що можуть, в якійсь мірі, частково захистити контент.

1. Спеціальні атрибути.

Можна застосувати атрибути, що забороняють копіювання. Якщо код написаний на JavaScript, можна прописати так (приклад коду на мові JavaScript):

<body oncopy = 'return false' oncut = 'return false'>

І користувачі не зможуть скопіювати текст у буфер обміну, але це дозволить виділення тексту.

Можна заборонити виділення тексту, прописавши:

<body onmousedown = 'return false' onselectstart = 'return false'>

Наступний код запобігає клацанню ПКМ на будь-якій сторінці. Треба захопити подію onContextMenu і в обробник події повернути false.

<body oncontextmenu="return false">

Існують і мінуси таких прийомів, адже часто користувачі виділяють абзаци чи періодично клацають ПКМ під час читання, просто за звичкою. І через це вони можуть зазнати дискомфорт і покинуть вебсайт, вважаючи недружнім.

Також можна прописати скрипт, щоб при копіюванні контенту в кінці тексту записувалось посилання на вебресурс. Звісно, від копіювання даний спосіб не захистить, але принесе користь, бо людина може не помітить посилання, розмістить у такому вигляді у себе і ваш вебсайт поповниться одним безкоштовним беклінком.

Дієвим способом є створення окремої сторінки, де необхідно прописати правила використання контенту вашого вебресурсу. Теоретично, людину, яка порушила правила буде чекати суд та штраф, але на практиці таке зустрічається рідко. Такий захист від крадіжки тексту являється надійним, бо прописані правила лякають копіювальників [9].

Висновки до розділу 2

Сучасний медіаконтент складається з різних елементів. Це відео, аудіо, картинки, ігри та інше. Такий вид контенту використовується для залучення користувачів та дає змогу широко користуватися можливостями інтернету.

Під час підготування інформації для вебсайтів слід враховувати його особливості, щоб всі елементи контенту співпадали з очікуваннями аудиторії та пробуджували інтерес.

Наприклад, опубліковану статтю на сайті можна доповнити фото й інфографікою чи провести опитування серед читачів на важливу тему, закликати до написання особистої думки в коментарях.

При додаванні тексту до вебсайту необхідно враховувати деякі особливості. Вебсередовище не має представлення традиційного газетного тексту з фіксованими розмірностями, початком та кінцівкою, а створює, в деякій мірі, нестабільний і динамічний вебтекст.

Колір на сайті має великий вплив на емоції людини. У мозку починають відбуватися складні процеси, які спочатку зчитують колір очима і, як підсумок, у людини виникає різний настрій. Чи успішне буде застосування тих чи інших колірних рішень, у великій мірі залежить від специфіки сайту, а також його цільової аудиторії.

3 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВЕБЗАСТОСУНКУ ІНТЕРНЕТ-ВИДАННЯ

3.1 Розробка варіантів використання. Діаграма прецедентів

Перед початком розробки додатку слід якісно і розгорнуто провести моделювання процесів ПЗ. Після опису специфікації та мети програми необхідно починати розробку діаграми використання системи, у якій буде зображено основні способи використання вебзастосунку.

Роздивимося декілька варіантів використання, які передбачає система, які описані в таблицях 3.1 – 3.5:

Таблиця 3.1 – Реєстрація читача

Назва	Реєстрація читача
Рівень	Мета користувача
Учасники	Читач
Зацікавлені сторони та інтереси	1. Читач: зацікавлений у пошуку та перегляду свіжих новин, на якому можна ділитися думками. . 2. Компанія (журналіст, медіа-проект, тощо): зацікавлені у збільшенні кількості зареєстрованих читачів.
Передумови	Читач шукає сайт з новинами, на якому можна ділитися думками.
Гарантія успіху	Читач: реєструється в системі та може висловлювати думки; Система: ідентифікувати користувача.

Кінець таблиці 3.1

Основний успішний сценарій	1. Користувач вирішує зареєструватися в системі та заходить на сторінку реєстрації. 2. Система запитує облікові дані користувача (ім’я, логін та пароль) для реєстрації. 3. Користувач вводить облікові дані для реєстрації, натискає кнопку “Реєстрація”. 4. Система перевіряє облікові дані користувача: логін (вірно записана електронна адреса) та пароль (складається з мінімум 8 символів, з 1 великою літерою та обов’язково наявність цифри). 5. Система вносить облікові дані користувача в базу даних та дозволяє вхід до системи.
Розширення сценаріїв	1) Користувач вводить невірну електронна адресу. 1. Система – повідомляє користувача про невірний логін. Пропонує користувачеві ввести вірний логін. 2. Користувач – повторює введення логіна. 3. Система – перевіряє логін, якщо вірно перейти до шагу 4, якщо ні – повернення до шагу 1. 4. Система – дозволяє вхід до сайта.

Таблиця 3.2 – Авторизація читача

Назва	Авторизація читача
Рівень	Мета користувача
Учасники	Читач

Кінець таблиці 3.2

Зацікавлені сторони та інтереси	1. Читач: зацікавлений у пошуку та перегляду свіжих новин. 2. Компанія (журналіст, медіа-проект, тощо): зацікавлені у збільшенні кількості зареєстрованих читачів.
Передумови	Читач шукає сайт з новинами, на якому можна ділитися думками.
Гарантія успіху	Читач: авторизуватися в системі та може висловлювати думки; Система: ідентифікувати користувача.
Основний успішний сценарій	1. Читач відкриває сайт. Система відкриває сесію користувача, пропонує ввести логін та пароль. 2. Користувач вводить ім’я, логін (електронна пошта) та пароль. 3. Система перевіряє логін та пароль – проходження валідації. 4. Система створює запис в історії авторизації (IP-адреса користувача, логін, дата). 5. Користувач успішно авторизований і може працювати із системою.
Розширення сценаріїв	1. Користувач забув пароль: 1.1. Система видає повідомлення. 1.2. Результат: користувач не може увійти. 2. Користувач не зареєстрований (Логін відсутній у базі даних): 2.1. Система видає повідомлення з проханням зареєструватися.

Таблиця 3.3 – Додавання статті

Назва	Авторизація читача
Рівень	Мета адміністратора
Учасники	Адміністратор
Зацікавлені сторони та інтереси	1. Адміністратор: додавання новин для оновлення інформації на сайті. 2. Читач: зацікавлені у регулярному оновленні інформації. 3. Компанія (журналіст, медіа-проект, тощо): зацікавлені у збільшенні кількості читачів.
Передумови	Читач шукає сайт з новинами, які додаються по мірі надходження інформації.
Гарантія успіху	Читач: зайшов на сайт, де відображені свіжі новини; Адміністратор: ідентифікував себе у системі, перейшов на панель адміністрування та додав статтю.
Основний успішний сценарій	1. Адміністратор відкриває сайт та переходить до сторінки входу до системи. 2. Адміністратор вводить логін та пароль. 3. Система перевіряє логін та пароль. 4. Система створює запис в історії авторизацій (IP-адреса користувача, логін, дата). 5. Адміністратор успішно авторизований і може працювати із системою. 6. Адміністратор переходить на сторінку «налаштування статті». 7. Адміністратор додає статтю та зберігає. 8. Адміністратор перевіряє, чи відображається нова стаття на сайті.

Кінець таблиці 3.3

Розширення сценаріїв	1. Адміністратор не може зберігти зміни: 1.1. Система видає повідомлення. 1.2. Результат: адміністратор не може зберігти статтю у БД.

Таблиця 3.4 – Пошук на сайті

Назва	Пошук на сайті
Рівень	Мета читача
Учасники	Читач
Зацікавлені сторони та інтереси	Читач: зацікавлений у швидкому пошуку потрібної інформації.
Передумови	Читач заходить на сайт та хоче швидко знайти інформацію на сайті.
Гарантія успіху	Читач: зайшов на сайт, ввів у полі пошуку та отримав результат у вигляді потрібної інформації;
Основний успішний сценарій	1. Читач відкриває сайт. 2. Користувач вводить у полі пошуку слово чи речення. 3. Система перевіряє введену інформацію з тою, що знаходиться на сайті. Якщо є збіг – виводиться стаття з шуканою інформацією, якщо немає – вивід усіх статей. 4. Користувач бачить результат та може продовжити роботу на сайті.
Розширення сценаріїв	1. Користувач не може ввести у поле «Пошук»: 1.1. Система не відповідає. 1.2. Результат: користувач не може знайти потрібну інформацію.

Таблиця 3.5 – Коментування статті

Назва	Коментування статті
Рівень	Мета читача
Учасники	Читач
Зацікавлені сторони та інтереси	Читачі: спілкування з іншими користувачами, обговорення статті.
Передумови	Читач має бути зареєстрований.
Гарантія успіху	Читач: зайшов на сайт, вибрав статтю та прокоментував.
Основний успішний сценарій	1. Читач відкриває сайт та переходить до сторінки входу до системи. Система пропонує ввести логін та пароль. 2. Читач вводить логін та пароль. 3. Система перевіряє логін та пароль. 4. Система створює запис в історії авторизацій (IP-адреса користувача, логін, дата). 5. Читач успішно авторизований і може працювати із системою. 6. Читач переходить на обрану статтю. 7. Читач додає коментар та зберігає.
Розширення сценаріїв	1. Читач не може зберігти коментар: 1.1. Система видає повідомлення. 1.2. Результат: читач не зареєстрований.

Було побудовано структурну схему використання (рис. 3.1).

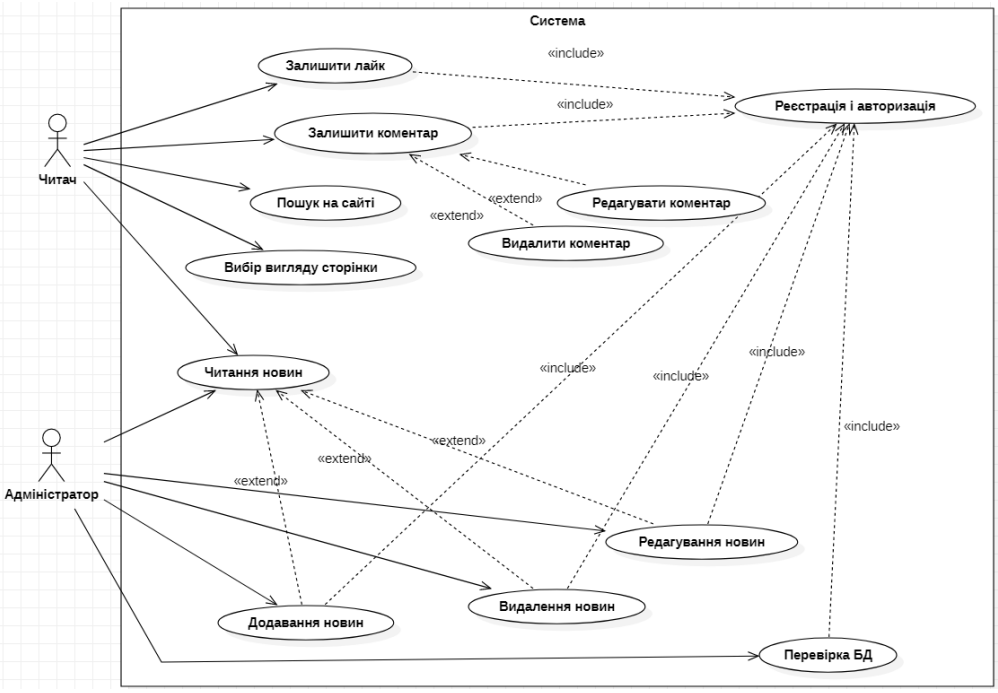


Рисунок 3.1 – Діаграма варіантів використання

Взявши за основу проаналізовану модель варіантів використання, ми побудували структурну схему використання, яка буде зображати основні способи використання застосунку.

3.2 Проектування інтерфейсу вебзастосунку інтернет-видання

В ході реалізації було розроблено дизайн вебсайту. На головній сторінці можна спостерігати наступну розмітку для функціональних вимог: головна (Новини М), пошук, вхід, як можна побачити на рисунку 3.2 – 3.4 для різних розширень екранів, так як сайт зроблено адаптивно до різних пристроїв.

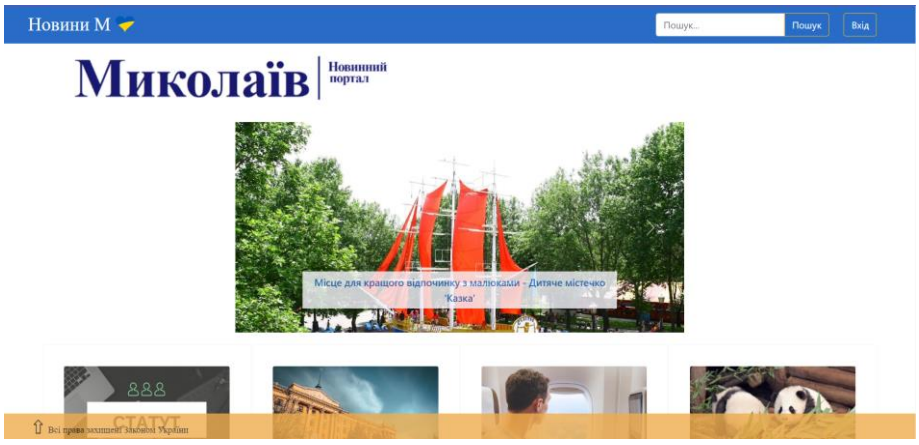


Рисунок 3.2 – Головна сторінка сайту (розширення 1180x820)

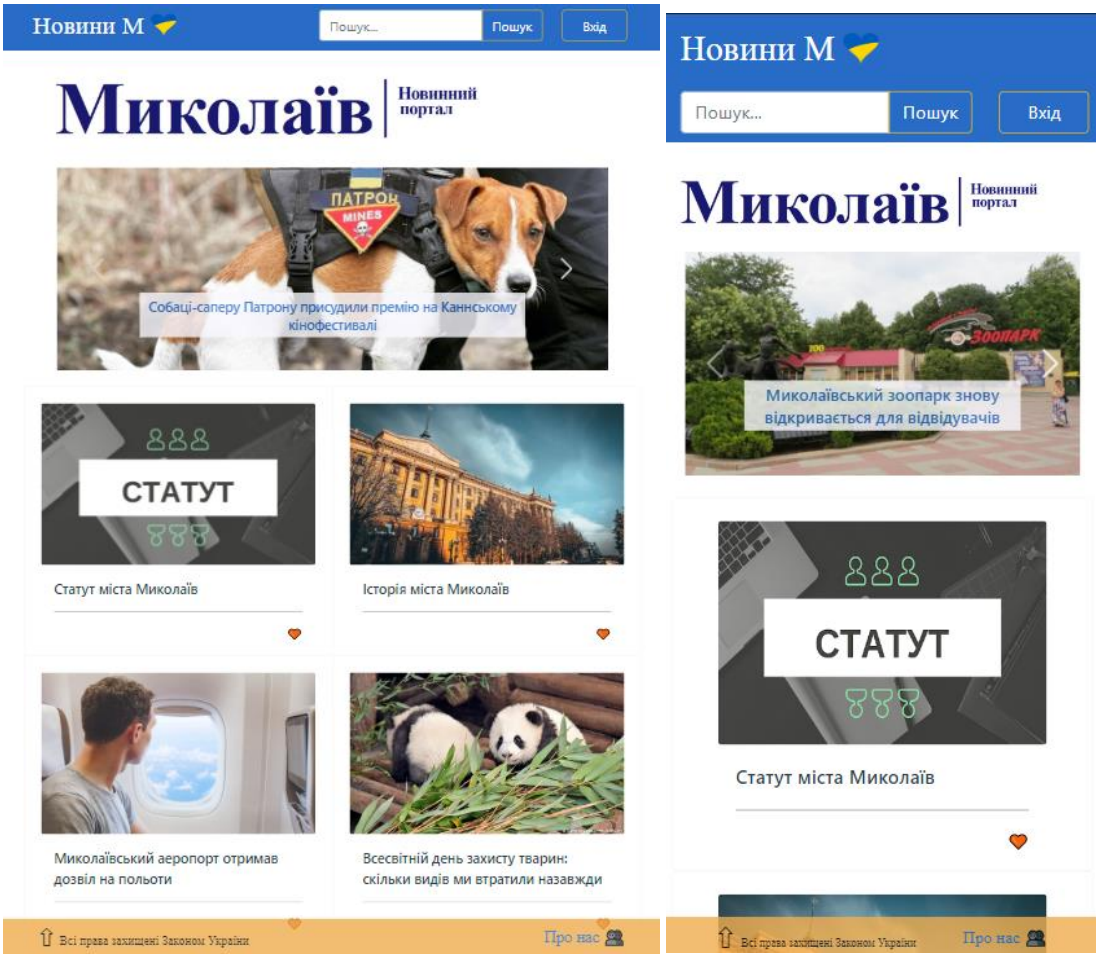


Рисунок 3.3 – 3.4 – Головна сторінка сайту
(розширення 820x1180, 390x844 відповідно)

Також вгорі знаходиться слайдер, на якому будуть відображатися останні три новини та статті з картинкою і заголовком.

Для досягнення адаптивності сайту буде використано Bootstrap Grid system. Система сіток використовує контейнери, рядки і стовпці (максимальна кількість 12) для компонування та вирівнювання вмісту по ширині екранів (табл. 3.6). Для адаптивності до ширини пікселів буде використано .container.

Таблиця 3.6 – Система сіток

Ширина контейнера	<576 пікселів	Невеликий ≥576 пікселів	Середній ≥768 пікселів	Великий ≥992 пікселів	Дуже великий ≥1200 пікселів
Префікс класу	.col-	.col-sm-	.col-md-	.col-lg-	col-xl-

Сторінки входу і реєстрації виглядають наступним чином (рис. 3.5 – 3.6):

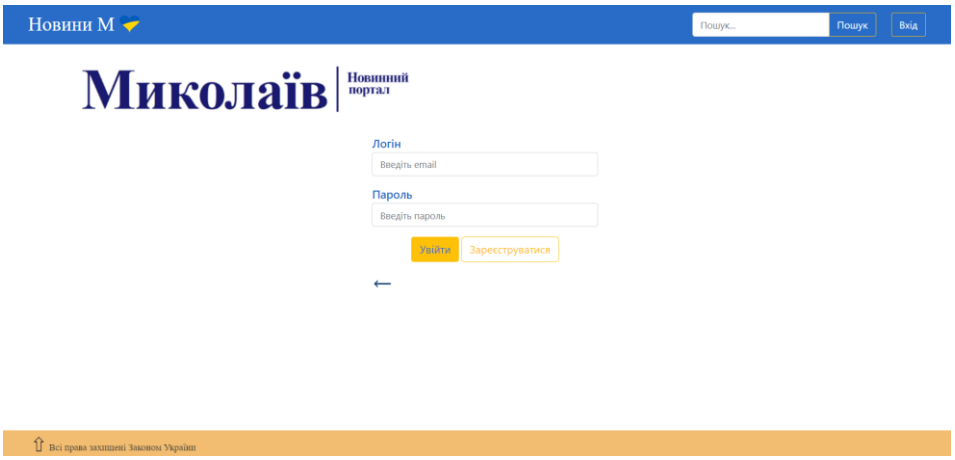


Рисунок 3.5 – Сторінка входу

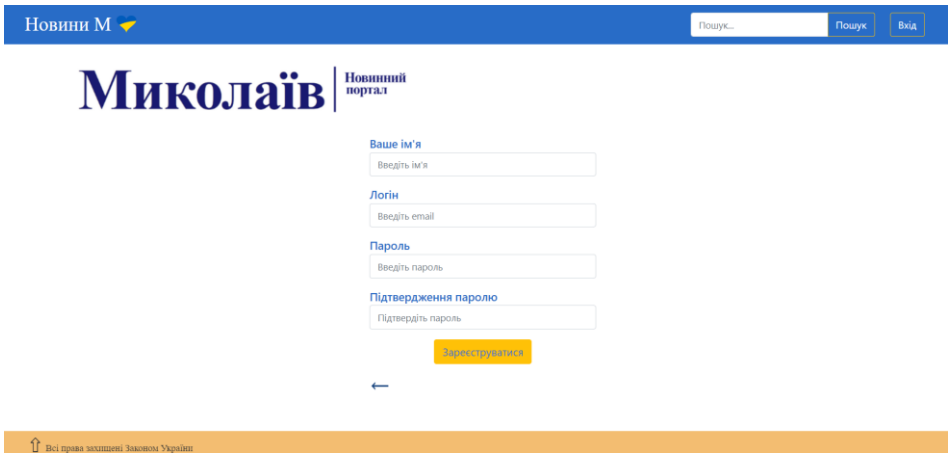


Рисунок 3.6 – Сторінка реєстрації

Сторінка пошуку має такий вигляд (рис. 3.7):

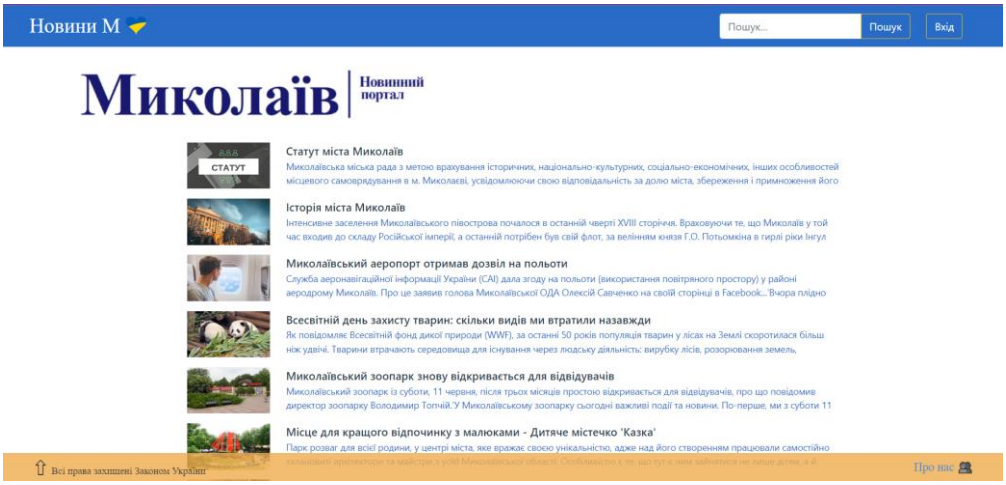


Рисунок 3.7 – Сторінка пошуку

У завданні до роботи було висунуто вимогу щодо дизайну статті. Користувач може вибирати зручний для себе дизайн сторінки. На сторінці статті, вгорі справа, розміщено 3 кнопки для вибору зовнішнього вигляду змісту статті, а внизу знаходиться поле для вводу та відправлення коментаря і їх відображення (рис. 3.8 – 3.10):

Рисунок 3.8 – Сторінка статті (перший вид) для екранів з розширенням 820x1180

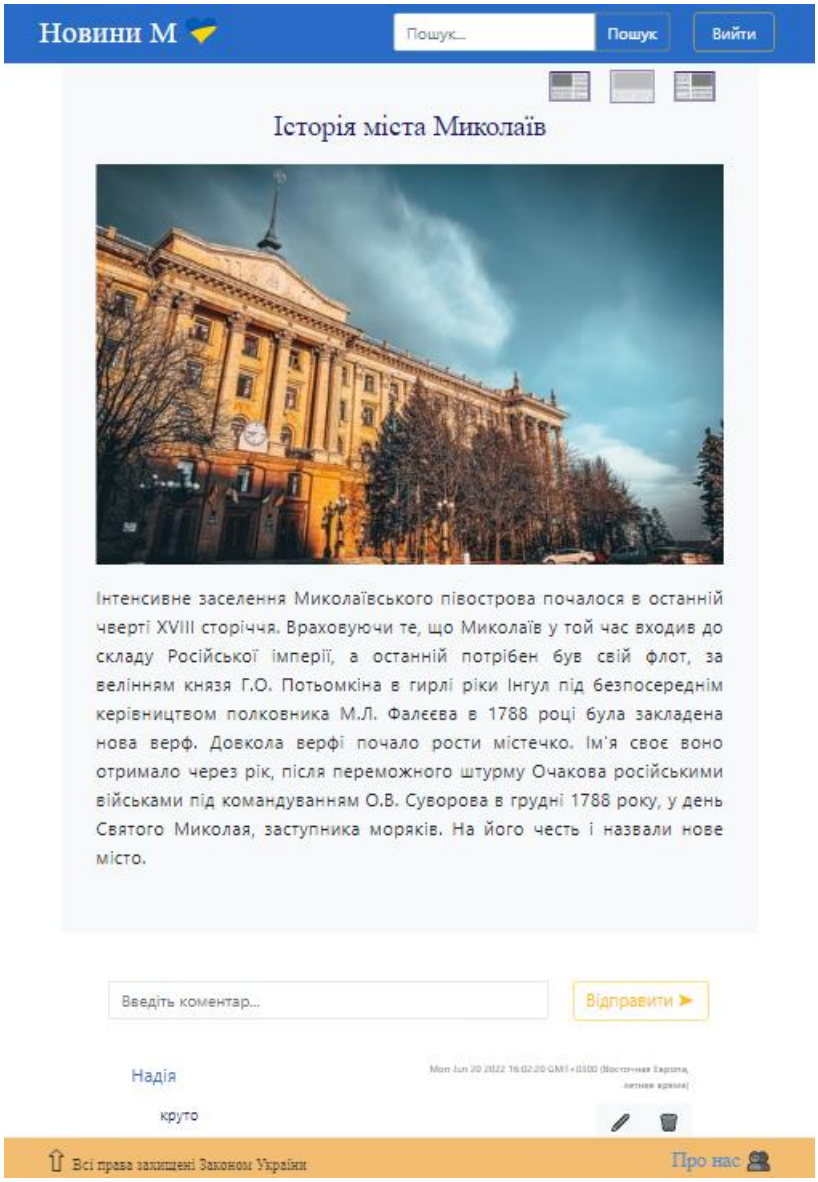


Рисунок 3.9 – Сторінка статті (другий вид) для екранів з розширенням 820x1180

Також біля кожного коментаря знаходитиметеся ім’я зареєстрованого користувача і інформація про час написання коментаря, а також кнопки для видалення та редагування тільки своїх коментарів.

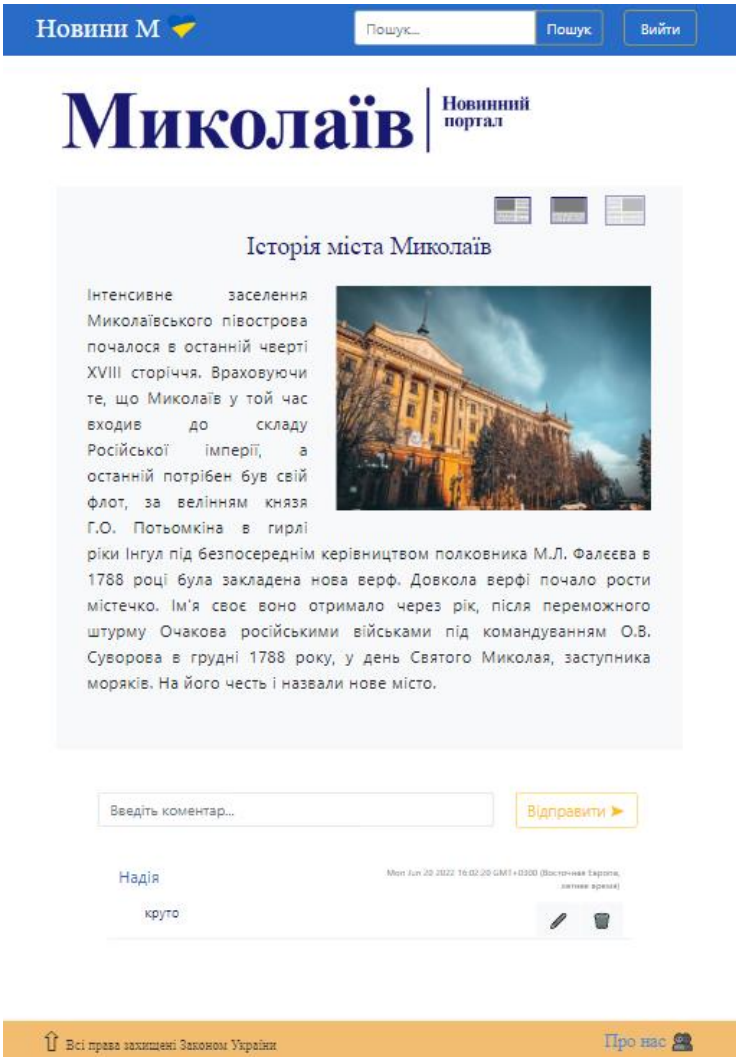


Рисунок 3.10 – Сторінка статті (третій вид) для екранів з розширенням 820x1180

Тепер перейдемо до панелі адміністратора. На першій сторінці відображено дві кнопки для переходу на налаштування статтей та перегляду журналу входів (рис. 3.11).

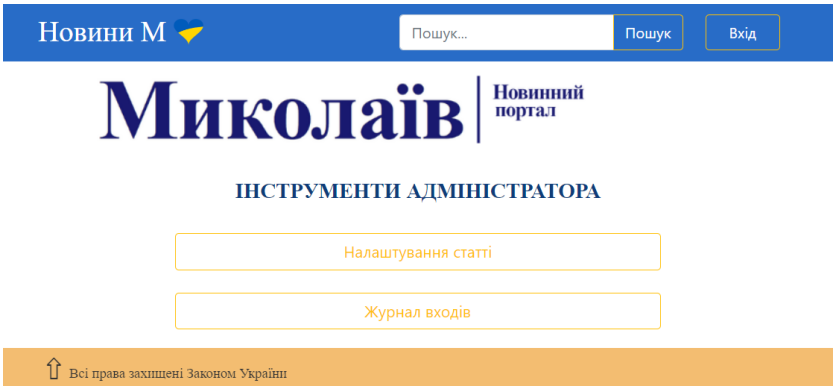


Рисунок 3.11 – Сторінка адміністратора (розширення 896x 414)

Сторінка налаштування статті має такий вигляд (рис. 3.12):

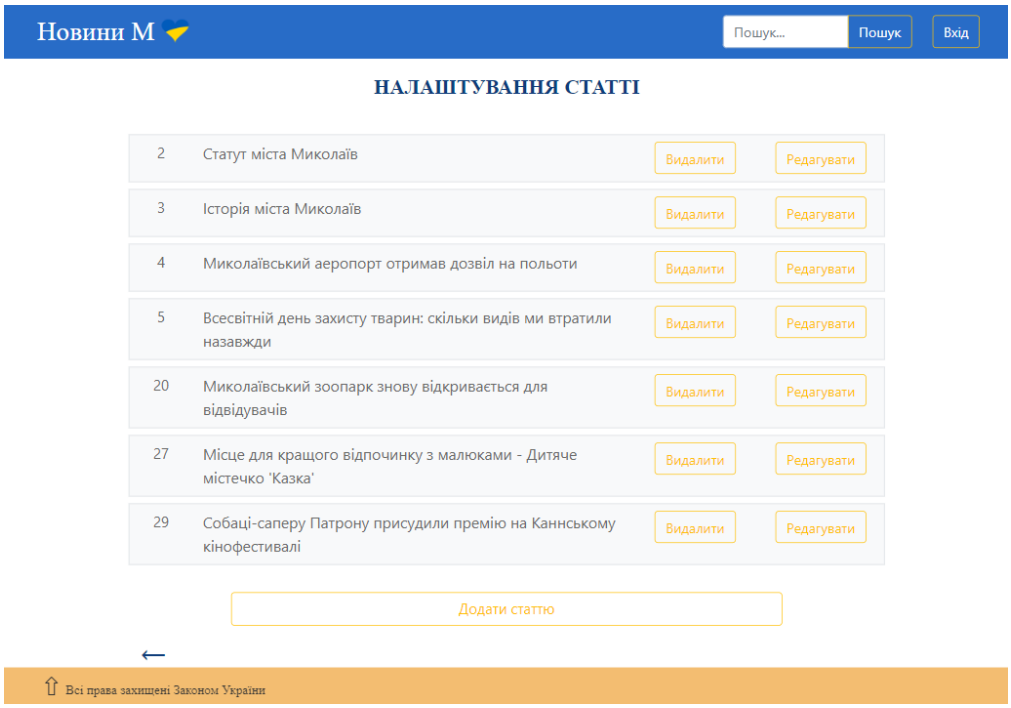


Рисунок 3.12 – Сторінка налаштування статті

Бачимо, що на цій сторінці є кнопка додавання статті, сторінка, на яку буде перехід по кнопці виглядає наступним чином (рис. 3.13):

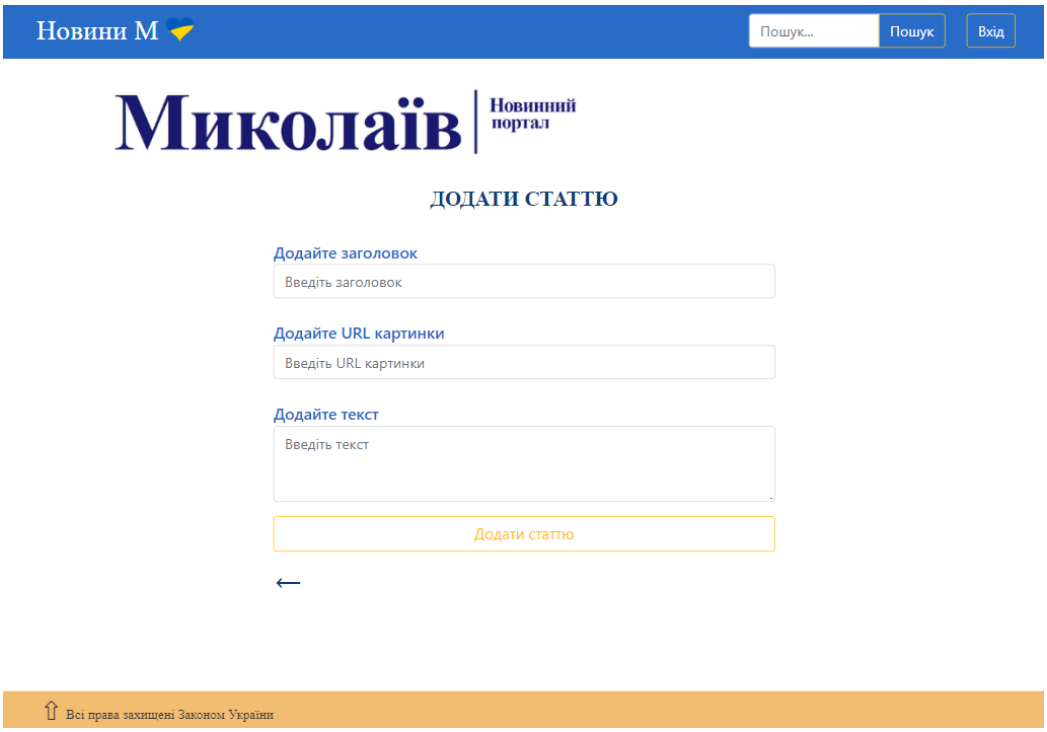


Рисунок 3.13 – Сторінка додавання статті

Для редагування статтей будемо використовувати ту ж сторінку, що і для додавання статтей, але змінимо назви полів для заповнення (рис. 3.14):

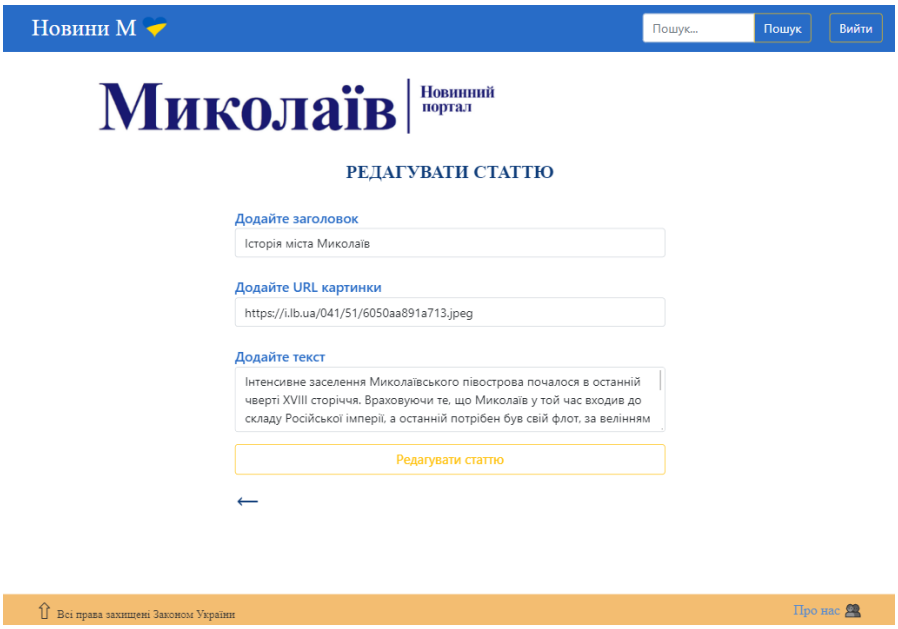


Рисунок 3.14 – Сторінка редагування статті

Журнал входів має наступний вигляд таблиці (рис. 3.15):

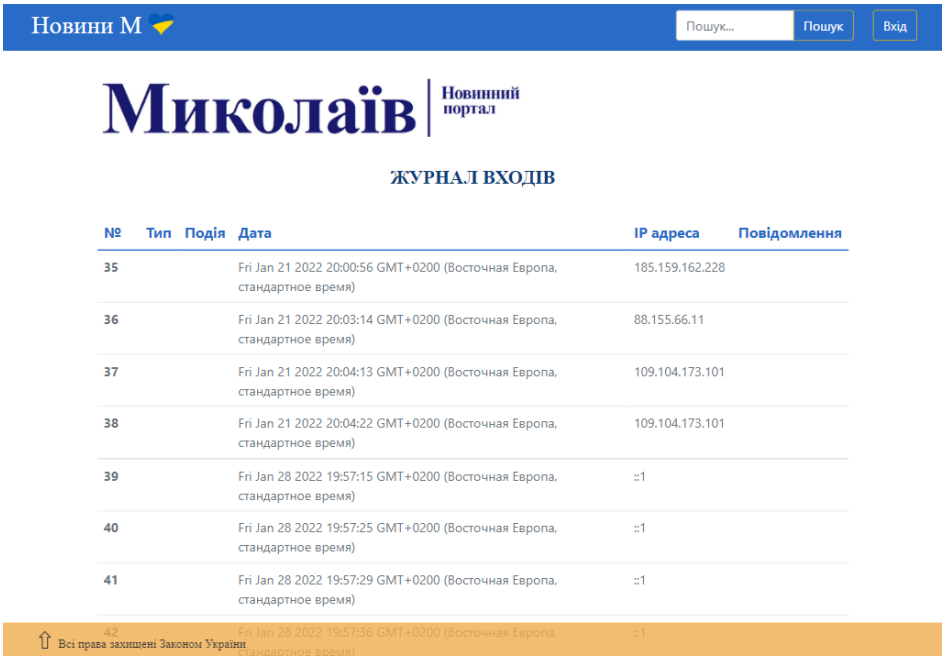


Рисунок 3.15 – Сторінка журналу входів

Для розробки будуть використовуватися бібліотека Bootstrap Vue та CSS.

3.3 Візуальна карта вебзастосунку інтернет-видання

Допоміжним елементом у плануванні є візуальна карта, яка використовується для аналізу та прототипування. Це допоможе краще зрозуміти майбутню навігацію [10] (рис. 3.16 – 3.17).

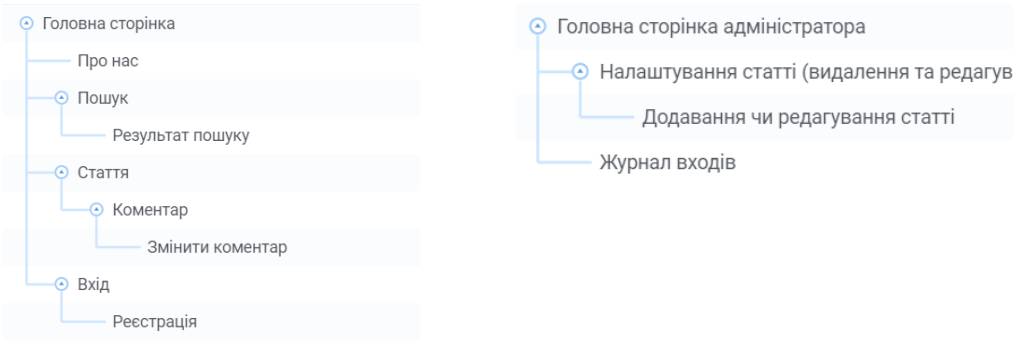


Рисунок 3.16 – Карта сторінок та переходів для читачів та адміністратора

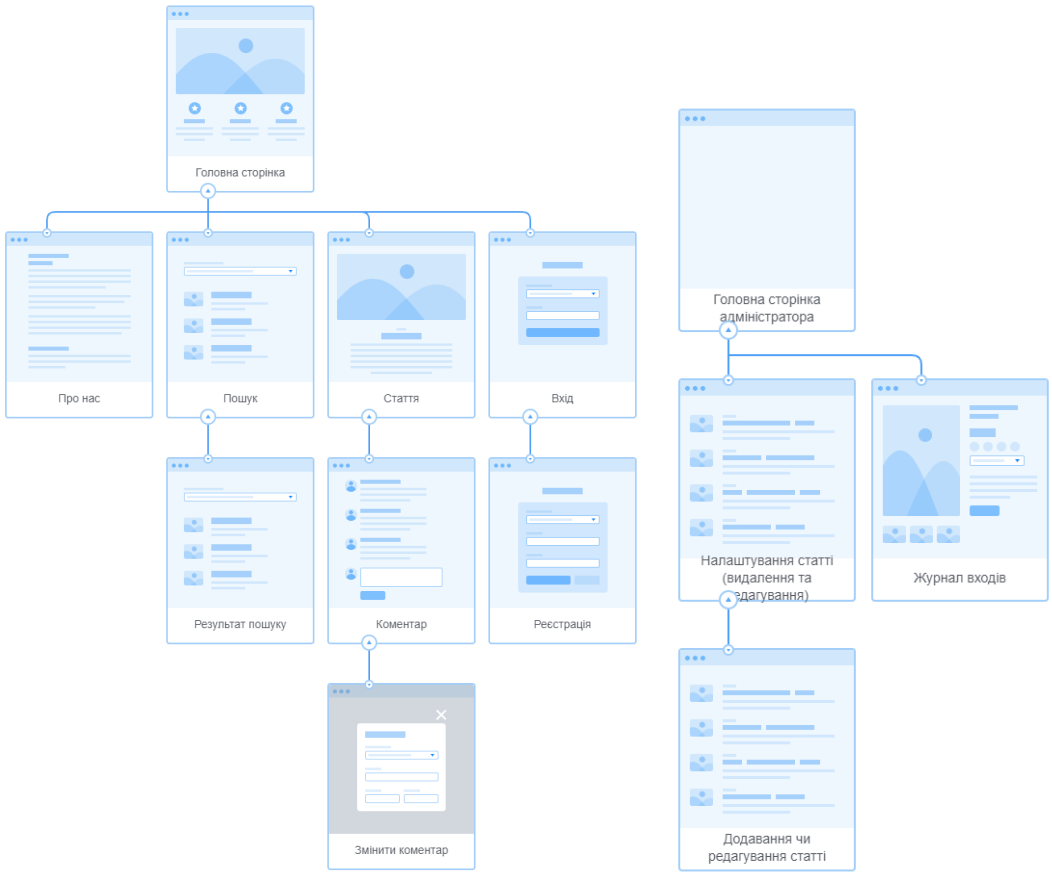


Рисунок 3.17 – Візуальна карта сторінок та переходів для читачів та адміністратора

Дані діаграми були розроблені на вебсервісі «flowmapp» [11].

3.4 Логічна модель бази даних веб-застосунку інтернет-видання

В процесі проектування виділено 5 головних об’єктів, серед яких користувач, стаття, коментар, вподобання та робочий журнал.

Використовуючи дану інформацію, було створено таблицю ідентифікаторів, яку представлено у таблиці 3.7.

Таблиця 3.7 – Таблиця ідентифікаторів

Об’єкт	Властивість	Тип і довжина	Призначення
Users	Id (Primary key)	Int, not null, autoincrement	Ідентифікатор
	UserName	Nvarchar (100) , not null	Ім’я користувача
	Email	Nvarchar (320) , not null	Електронна пошта
	Passwords	Nvarchar (320) , not null	Пароль
WorkLog	Id (Primary key)	Int, not null, autoincrement	Ідентифікатор
	LogTypeId	Int, not null	Тип
	LogEventId	Int, not null	Подія
	Message	Varchar (100) , null	Повідомлення
	Date	Datetime, not null	Дата і час
	IpAddress	Varchar (50) , null	Айпі адреса
Article	Id (Primary key)	Int, not null, autoincrement	Ідентифікатор
	Title	Nvarchar (100) , null	Заголовок
	Image	Nvarchar (max) , null	Картинка
	Text	Nvarchar (100) , null	Текст
Comment	Id (Primary key)	Int, not null, autoincrement	Ідентифікатор
	UserId	Int, not null	Id користувача
	ArticleId (FK)	Int, not null	Id статті
	DateCreated	Datetime, not null	Дата створення
	Comment	Nvarchar (200) , not null	Коментар

Кінець таблиці 3.7

Like	Id (Primary key)	Int, not null, autoincrement	Ідентифікатор
	UserId (FK)	Int, not null	Id користувача
	ArticleId (FK)	Int, not null	Id статті

Було побудовано ER-діаграму бази даних згідно таблиці, яку можна побачити на рисунку 3.18.

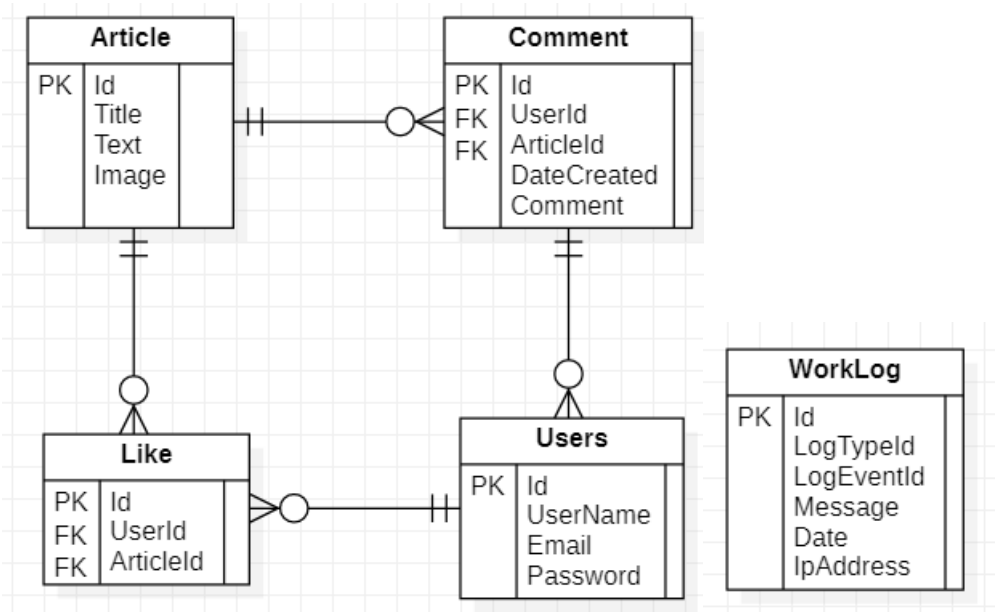


Рисунок 3.18 – ER-діаграма бази даних

Висновки до розділу 3

В даному розділі було виконане моделювання та проектування програмного забезпечення вебзастосунку інтернет-видання. Побудовано структурну схему використання, яка зображує основні способи використання застосунку.

Спроектовано інтерфейс вебсервісу інтернет-видання, а також розроблена візуальна карта сторінок та переходів для користувача та адміністратора. Було створено таблицю ідентифікаторів і розроблено ER-діаграму бази даних, яка являє собою 5 таблиць, в яких буде зберігатися основна інформація порталу.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ІНТЕРНЕТ-ВИДАННЯ

4.1 Вибір технологій розробки та архітектура вебзастосунку інтернет-видання

Для розробки клієнтської сторони інтерфейсу було використано такі технології, як HTML, CSS, Vue.js та Bootstrap Vue.

Для програмно-апаратної частини сторони було обрано мову програмування C#. Середою розробки обрано Visual Studio 2019, а платформою – .Net Core, для кросплатформності та забезпечення максимально швидкої і простої розробки.

Для роботи з сайтом було використано СУБД MSSQL Server та мову T-SQL. Це система управління БД, яка була розроблена корпорацією Microsoft [14]. SQL Server має такі особливості як продуктивність, надійність, безпека та простота.

Для зв’язку з базою даних використано Dapper. Він внутрішньо використовує ADO.NET. Dapper є інструментом, що зіставляє результати sql-запитів із класами C#. За рахунок легкості, Dapper гарантує високу продуктивність і дозволяє швидко обробляти та виконувати запити. При роботі з Dapper використовується функціонал пакету Microsoft.Data.SqlClient, тому було додано пакети Microsoft.Data.SqlClient та Dapper [12].

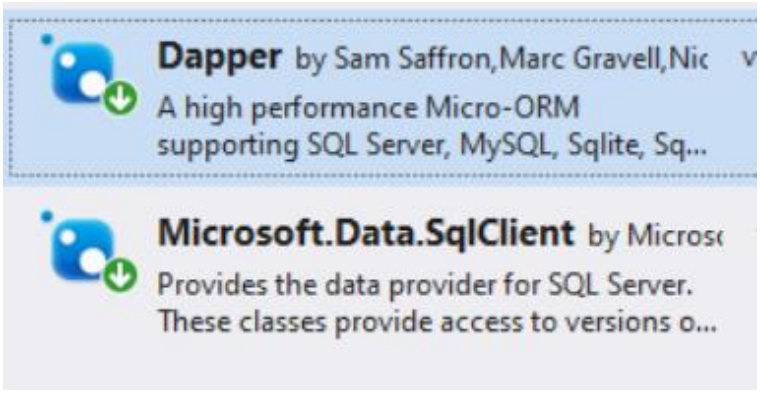


Рисунок 4.1 – Пакети Microsoft.Data.SqlClient та Dapper

.NET Framework – це повнофункціональна платформа Windows. .NET Core – це модульна платформа для розробки програмного забезпечення з

відкритим вихідним кодом [13]. Дана платформа працює з такими операційними системами, як Windows, Linux і macOS.

У реалізації вебзастосунку було використано трирівневу архітектуру розробки (рис.4.2).

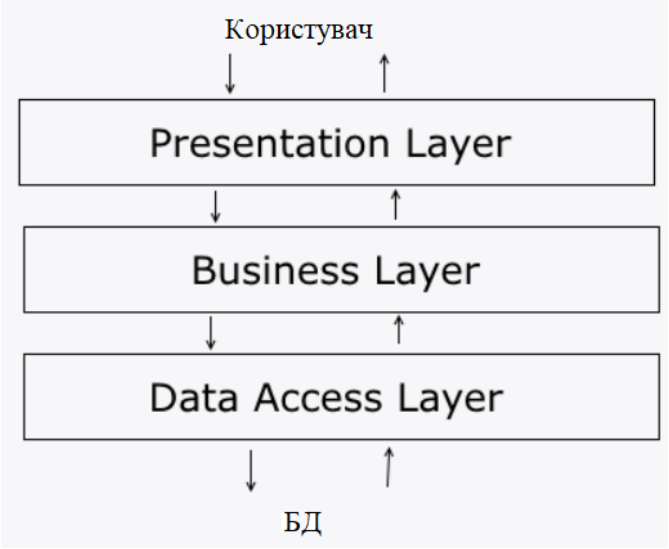


Рисунок 4.2 – Трирівнева архітектура розробки

Архітектура проекту наведена на рисунку 4.3.

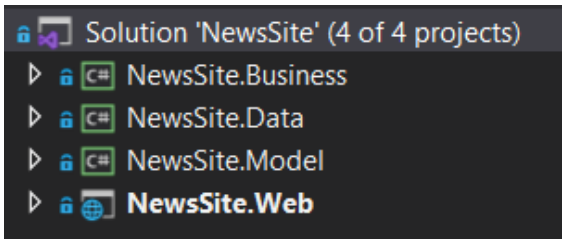


Рисунок 4.3 – Трирівнева архітектура проекту

1. Рівень представлення (Presentation layer).

Рівень, з яким взаємодіє користувач. Включає в себе компоненти інтерфейсу користувача, механізм отримання даних, які вводить користувач. На даному рівні розташовані всі компоненти, що відносяться до інтерфейсу користувача, а саме стилі, сторінки html та JavaScript, а також контролери, об'єкти запиту та результатів їх обробки.

2. Бізнес-логіка (Business layer).

Містить компоненти, які обробляють отримані дані від рівня представлення, реалізує логіку програми, обчислення, взаємодіє з БД і віддає результат обробки рівню представлення.

3. Рівень доступу до даних (Data Access layer).

Тут зберігаються репозиторії, якими рівень бізнес-логіки взаємодіє з базою даних.

Треба зауважити, що граничні рівні не можуть взаємодіяти між собою. Інакше кажучи, рівень представлення не може звертатися до БД і рівня доступу до даних. Він може взаємодіяти тільки через рівень бізнес-логіки. Рівень доступу до даних напряму має залежність від інших рівнів.

Весь шлях обробки інформації та роботи архітектури показано на рисунку 4.4.

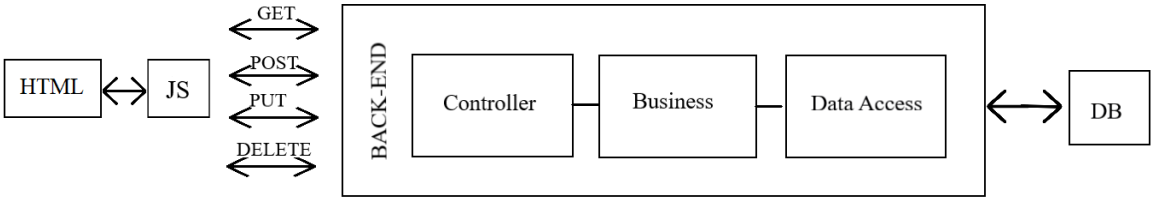


Рисунок 4.4 – Взаємодія між Front-End та Back-end

За передачу запитів і отримання результатів обробки цих запитів відповідає протокол HTTP (протокол передачі гіпертекстів). HTTP є протоколом типу запит / відгук, клієнт відправляє серверу запит. Визначено такі методи протоколу – GET, POST, HEAD, PUT, DELETE, TRACE та OPTION. Найбільш часто використовуються перші два методи [15].

Для запиту конкретного ресурсу використовують метод GET. Параметри запиту передають в URL у вигляді пар, за допомогою знаків '?', = та '&'. POST використовують для передачі даних сервера в тілі запиту. PUT і DELETE рекомендовані для зміни ресурсів.

4.2 Діаграма класів

Діаграма класів – представлення структури моделі системи. Відображає взаємозв'язки між сутностями (об'єктами і підсистемами), і описує внутрішню структуру. Графічно клас зображують у вигляді прямокутника, який може бути розділений лініями на розділи, такі як ім’я класу, атрибути та операції.

Графічне зображення класу на діаграмі класів має наступний вигляд (рисунок 4.5) [16].

Рисунок 4.5 – Графічне зображення класу

Діаграма класів розроблена у Visual Studio 2019. Так як проект вийшов досить великий, діаграми класів наведені для рівнів проекту. На рисунку 4.6 зображено рівень представлення.

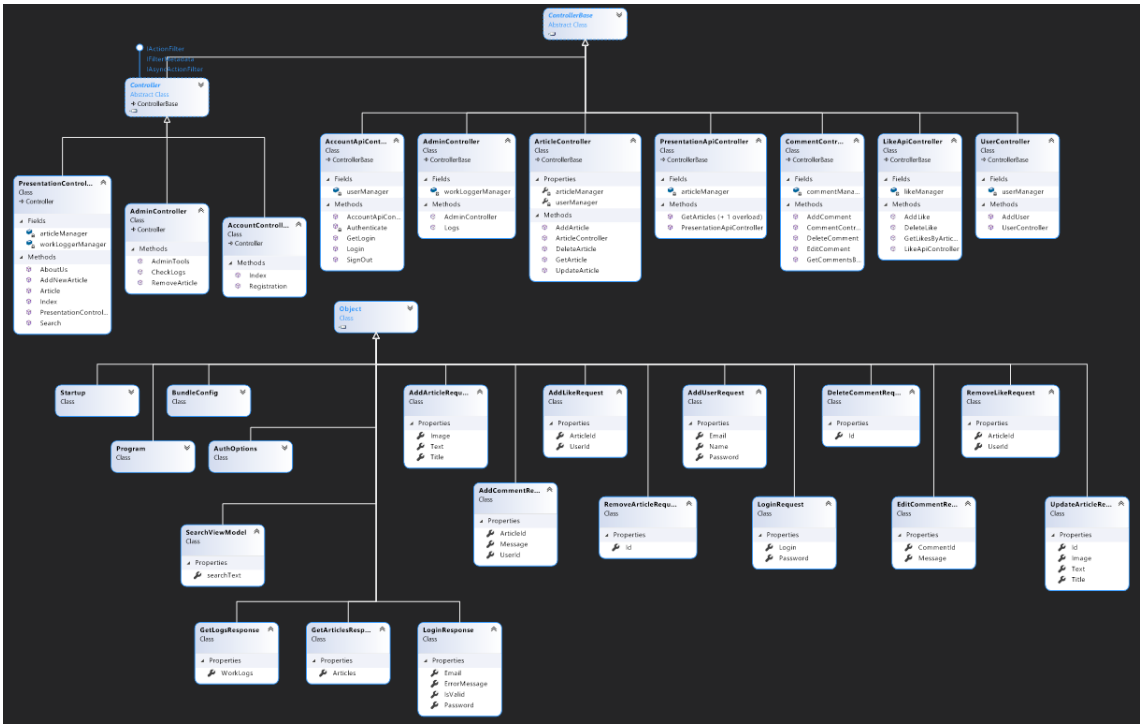


Рисунок 4.6 – Діаграма класів для рівня представлення

Роздивимося поближче окремі елементи даної діаграми На рисунку 4.7 зображено MVC контролери, які одержують вхідні дані та перетворюють їх на команди для моделі чи вигляду.

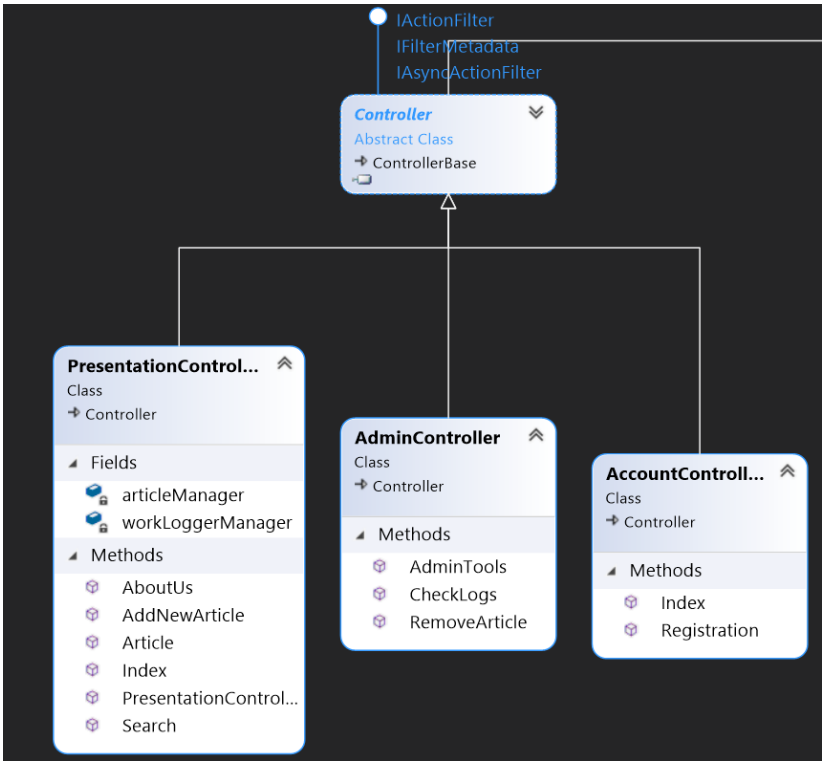


Рисунок 4.7 – Діаграма класів для рівня представлення (MVC контролери)

API контролери, які обробляють запити HTTP і надсилають відповідь зображено на рисунку 4.8.

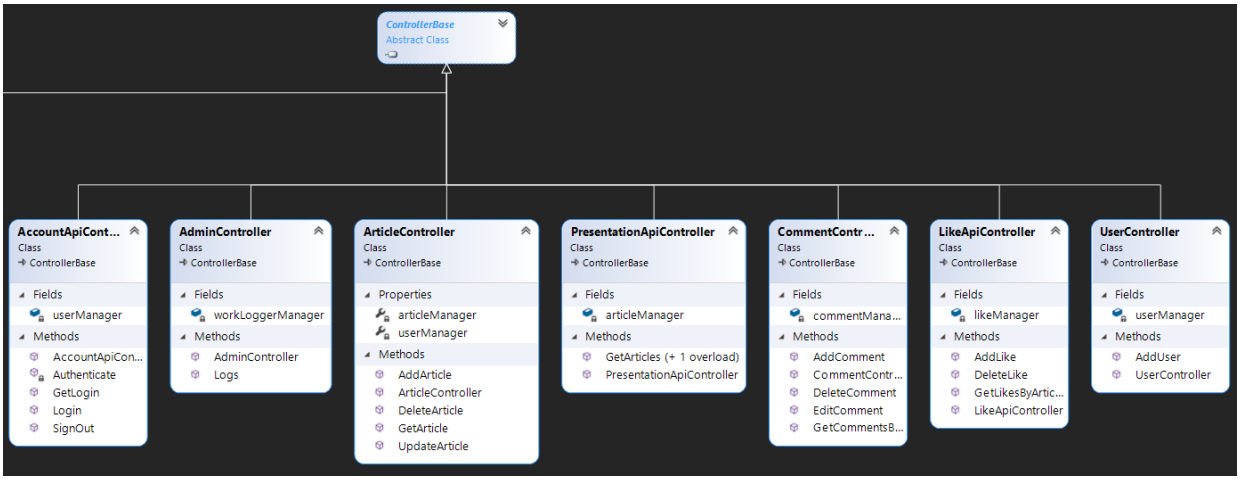


Рисунок 4.8 – Діаграма класів для рівня представлення (API контролери)

На рисунку 4.9 зображено бізнес рівень.

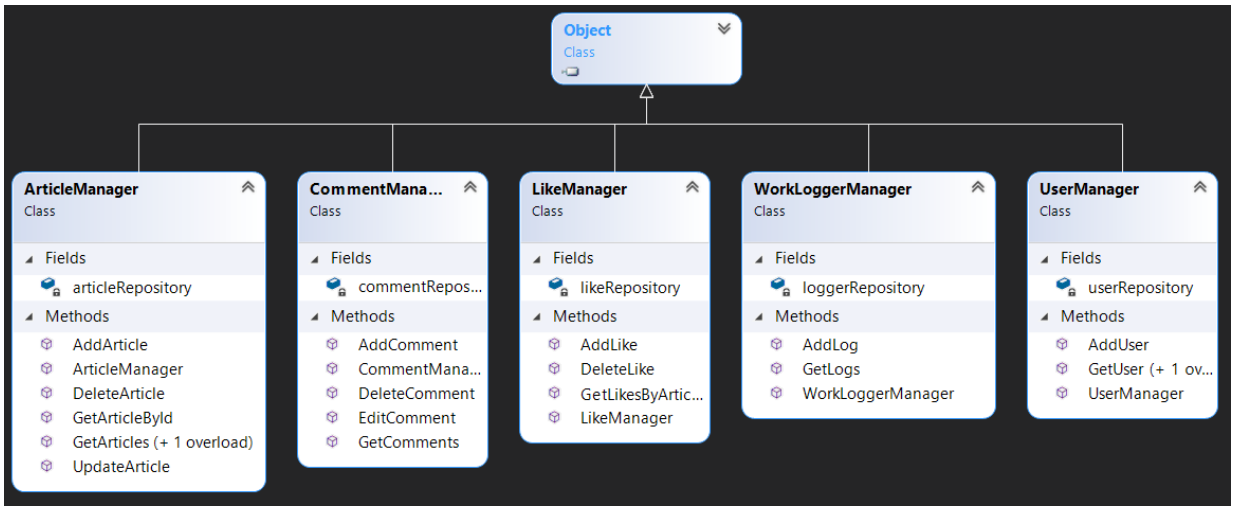


Рисунок 4.9 – Діаграма класів для бізнес рівня

На рисунку 4.10 відображено моделі БД, які були винесені в окрему папку проекту задля легшого звернення до них.

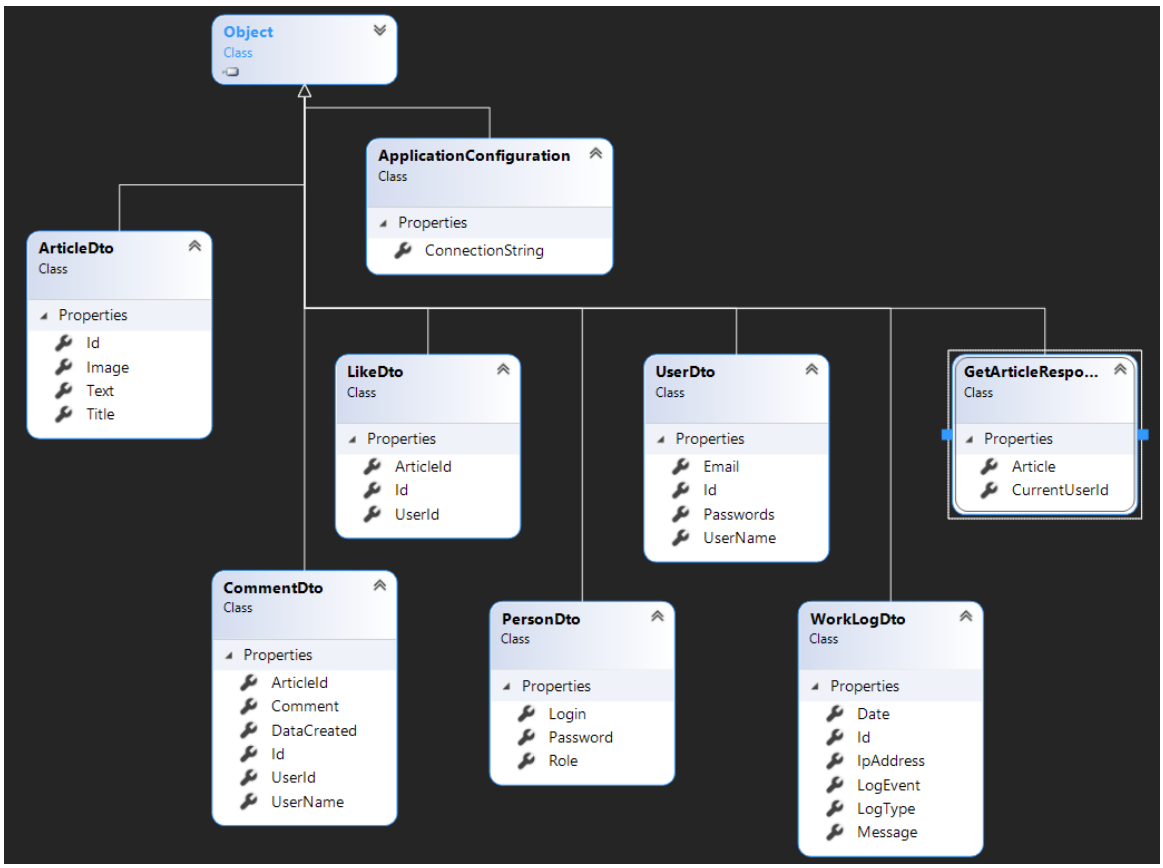


Рисунок 4.10 – Діаграма класів для моделей БД

На наступному рисунку 4.11 відображено класи рівня доступу до даних, тобто репозиторії.

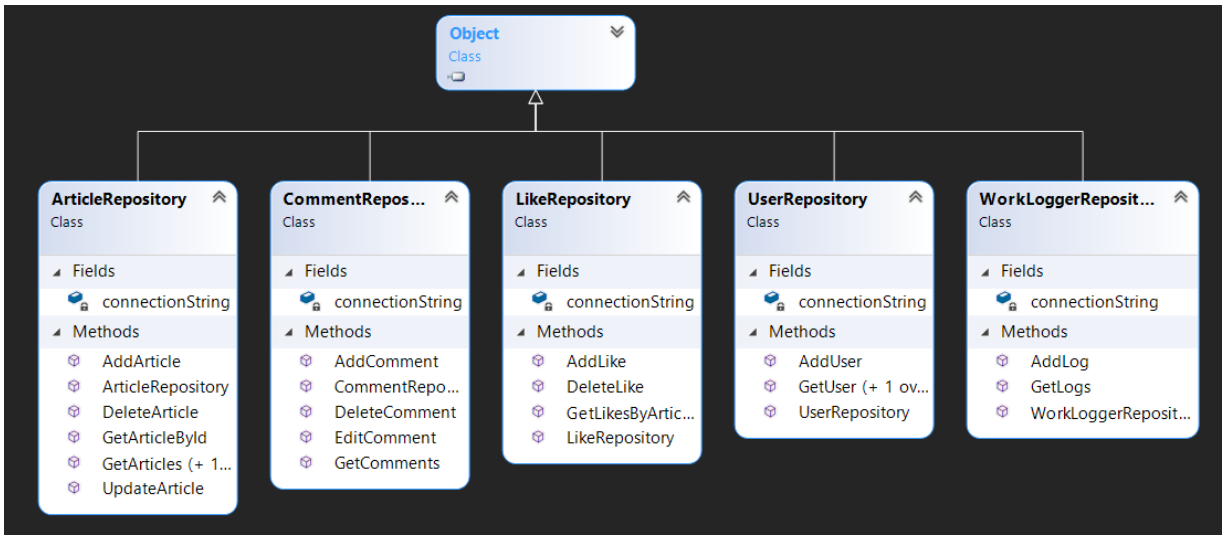


Рисунок 4.11 – Діаграма класів рівня доступу до даних

Діаграма класів дозволила представити логічну модель програми на рівні структури, тобто класів. Дозволяє показати з яких класів складається проект та які зв’язки мають класи і з чого складаються класи.

4.3 Розробка бази даних

Так як у розробці використовується трирівнева архітектура спочатку відбувається звернення до контролеру, який має певний набір дій. Далі звернення до окремої дії, яка в свою чергу викликає модель й отримує дані для подальшої обробки.

Для того, щоб звертатися до таблиць у БД треба визначити об’єкти передачі даних (DTO). DTO – це об’єкт (клас), що визначає спосіб відправлення даних, по іншому це називають модель і успадковується від класу Dtos, який наслідується від класу Model [17].

Приклад створених у проекті моделей відповідних до таблиць БД (рис. 4.12).

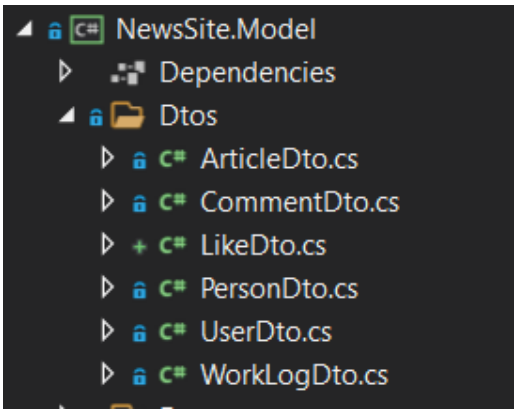


Рисунок 4.12 – Зображення папки DTO

Для реалізації завдання нам потрібно спроектувати відповідну БД. Було створено 5 таблиць, а саме таблиці Article – у якій будуть зберігатися всі статті, User – дані про регістрацію користувача, WorkLog – дані про входження на сайт, Comment – зберігаються коментарі користувачів та Like – вподобання користувачів статтей.

Для створення таблиць було створено запит на мові T-SQL. Наведемо приклад створення таблиці.

```
CREATE TABLE table_name (
    pk_column data_type PRIMARY KEY, KEY IDENTITY (),
    column_1 data_type NOT NULL,
    column_2 data_type,
    ...,
    table_constraints
);
```

Під час роботи з БД було використано операції CRUD – Create, Read, Update, Delete. Вони використовуються у рівні доступу до даних. Приклад додавання статті до БД:

```
public void AddArticle(string title, string text, string image)
    {
       using (IDbConnection db = new SqlConnection(connectionString))
       {
          db.Query<ArticleDto>("INSERT INTO Article (Title, Image, Text) values (@title, @image , @text)", new {title = title, image = image, text = text});
       }
    }
```

4.4 Тестування вебзастосунку інтернет-видання

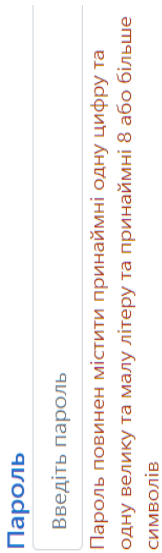
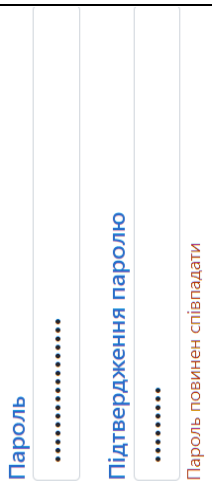
Кінцевим етапом розробки вебсайту є тестування. Тестування – це перевірка ПЗ на відповідність до визначених вимог. Важливим етапом в розробці є етап тестування. Вебсайт має постійну взаємодію з користувачем, тому слід бути впевненим, що процес буде відповідати заданим вимогам.

Тестові сценарії були розроблені на основі функціональних вимог, які описані в таблицях 4.1 – 4.7.

Таблиця 4.1 – Тест-кейс реєстрації

№	Назва	Передумова	Кроки	Очікуваний результат	Фактичний результат
1	Правильна реєстрація.	Відкрити сайт і форму реєстрації.	1. Ввести ім’я. 2. Ввести вірну пошту. 3. Ввести вірний пароль. 4. Підтвердити пароль. 5. Натиснути кнопку «Реєстрація».	Користувач отримує повідомлення про успішну реєстрацію і система перенаправляє на сторінку «Логін».	Користувач побачив повідомлення і перенаправлений на сторінку «Логін».

Продовження таблиці 4.1

2	Реєстрація з невірною поштою.	Відкрити сайт і форму реєстрації.	1.Ввести ім’я. 2.Ввести невірну пошту. 3.Ввести вірний пароль. 4.Підтвердити пароль. 5.Натиснути кнопку «Реєстрація».	Користувач отримує повідомлення про невірно вказану електронну пошту.	
3	Реєстрація з невірним паролем.	Відкрити сайт і форму реєстрації.	1.Ввести ім’я. 2.Ввести вірну пошту 3.Ввести невірний пароль. 4.Підтвердити пароль. 5.Натиснути кнопку «Реєстрація».	Користувач отримує повідомлення про невірний пароль.	
4	Реєстрація з невірним підтверд-женням пароля.	Відкрити сайт і форму реєстрації.	1.Ввести ім’я. 2.Ввести вірну пошту. 3.Ввести вірний пароль. 4.Не вірно підтвердити пароль. 5.Натиснути кнопку «Реєстрація».	Користувач отримує повідомлення про невірне підтвердження пароля.	

Кінець таблиці 4.1

5	Реєстрація з пустими полями.	Відкрити сайт і форму реєстрації.	1.Залишити поля не заповненими. 2.Натиснути кнопку «Реєстрація».	Користувач отримує повідомлення про невірну реєстрацію.	Користувач побачив повідомлення про невірну реєстрацію.

Таблиця 4.2 – Тест-кейс авторизації

№	Назва	Передумова	Кроки	Очікуваний результат	Фактичний результат
1	Правильна авторизація.	Користувач зареєстрований у системі. Відкрити сайт, форму авторизації.	1. Ввести вірну пошту. 2. Ввести вірний пароль. 3. Натиснути кнопку «Увійти».	Система перенаправляє користувача на головну сторінку і кнопка змінюється на «Вихід».	Користувач перенаправлений на головну сторінку і бачe кнопку «Вихід».
2	Авторизація з невірною поштою.	Впевнитися що користувача немає у системі.	1. Ввести невірну пошту. 2. Ввести вірний пароль. 3. Натиснути кнопку «Увійти».	Користувач отримує повідомлення про невірно вказану електронну пошту.	
3	Авторизація з невірним паролем.	Впевнитися що користувача немає у системі.	1. Ввести вірну пошту. 2. Ввести невірний пароль. 3. Натиснути кнопку «Увійти».	Користувач отримує повідомлення про невірний пароль.	

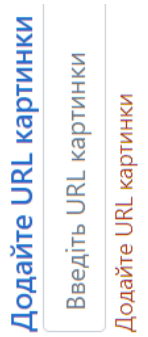
Кінець таблиці 4.2

4	Авторизація з пустими полями.	Відкрити сайт і форму авторизації.	1. Залишити поля не заповненими. 2. Натиснути кнопку «Увійти».	Користувач отримує повідомлення про невірну авторизацію.	Користувач побачив повідомлен-ня про невірну авторизацію.

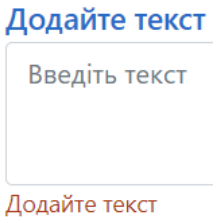
Таблиця 4.3 – Тест-кейс пошуку на сайті

№	Назва	Передумова	Кроки	Очікуваний результат	Фактичний результат
1	Пошук на сайті.	Відкрити сайт.	1. Вибрати слово з заголовку статті. 2. Ввести у пошук. 3. Перевірити, що дана стаття присутня у відповіді пошуку. 4. Натиснути кнопку «Пошук».	Система дає правильну відповідь на пошук.	Правильно виданий пошук.
2	Пошук на сайті (при різних регістрах до одного запиту).	Відкрити сайт.	1. Ввести слово у нижньому регістрі. 2. Ввести таке саме слово у нижньому регістрі. 3. Перевірити, що запит приводить до одної відповіді.	Система дає правильну відповідь на пошук з різними регістрами.	Правильно виданий пошук.

Таблиця 4.4 – Тест-кейс додавання статті

№	Назва	Передумова	Кроки	Очікуваний результат	Фактичний результат
1	Додавання статті.	Відкрити сайт, перейти на адмін-панель і натиснути на кнопку додавання статті.	1. Ввести заголовок. 2. Додати URL картинки. 3. Ввести текст статті. 4. Натиснути кнопку «Додати статтю».	Адміністратор отримує повідомлення про успішне додавання статті та система перекидує на сторінку перегляду всіх статтей.	Адміністратор побачив повідомлення про успішне додавання статті та був перекинутий на сторінку перегляду всіх статтей.
2	Додавання статті без заголовка.	Відкрити сайт, перейти на адмін-панель і натиснути на кнопку додавання статті.	1. Додати URL картинки. 2. Ввести текст статті. 3. Натиснути кнопку «Додати статтю».	Адміністратор отримує повідомлення про пустий заголовок.	
3	Додавання статті без URL картинки.	Відкрити сайт, перейти на адмін-панель і натиснути на кнопку додавання статті.	1. Ввести заголовок. 2. Ввести текст статті. 3. Натиснути кнопку «Додати статтю».	Адміністратор отримує повідомлення про пустий URL.	

Кінець таблиці 4.4

4	Додавання статті без тексту.	Відкрити сайт, перейти на адмін-панель і натиснути на кнопку додавання статті.	1. Ввести заголовок. 2. Додати URL картинки. 3. Натиснути кнопку «Додати статтю».	Адміністратор отримує повідомлення про пустий текст.	
5	Додавання пустої статті.	Відкрити сайт, перейти на адмін-панель і натиснути на кнопку додавання статті.	1. Залишити поля не заповненими. 2. Натиснути кнопку «Додати статтю».	Адміністратор отримує повідомлення про пусті поля.	Адміністратор отримав повідомлення про пусті поля.

Таблиця 4.5 – Тест-кейс додавання коментаря

№	Назва	Передумова	Кроки	Очікуваний результат	Фактичний результат
1	Додавання коментаря.	Користувач повинен бути зареєстрований у системі.	1. Обрати статтю. 2. Заповнити поле для комента. 3. Натиснути кнопку «Відправити».	Сторінка оновлюється та з’являється новий коментар.	Читач бачить свій коментар.
2	Додавання пустого коментаря.	Користувач повинен бути зареєстрований у системі.	1. Обрати статтю. 2. Натиснути кнопку «Відправити».	Повідомлення про помилку.	Читач бачить повідомлення.

Таблиця 4.6 – Тест-кейс редагування коментаря

№	Назва	Передумова	Кроки	Очікуваний результат	Фактичний результат
1	Редагування коментаря.	Користувач повинен бути зареєстрований у системі.	1. Обрати статтю. 2. Натиснути на кпопку зі значком ручка. 3. У модульному вікні змінити коментар. 4. Натиснути кнопку «Зберігти».	Сторінка оновлюється та з'являється новий коментар.	Читач бачить свій коментар.
2	Зміна на пустий коментар.	Користувач повинен бути зареєстрований у системі.	1. Обрати статтю. 2. Натиснути на кпопку зі значком ручка. 3. У модульному вікні змінити коментар на пустий. 4. Натиснути кнопку «Зберігти».	Користувач не може зберігти зміни.	Користувач не може зберігти зміни.

Таблиця 4.7 – Тест-кейс вподобання статті

№	Назва	Передумова	Кроки	Очікуваний результат	Фактичний результат
1	Вподобання статті.	Користувач повинен бути зареєстрований у системі.	1. Обрати статтю. 2. Натиснути на «сердце».	Біля «сердця» додається 1 до кількості лайків.	Читач бачить +1 у кількості лайків.

Було використано такий метод тестування, як «чорна скриня». Тестування таким методом також знайоме як тестування, що ґрунтується на специфікації або тестуванні поведінки. Це техніка тестування, яка використовується у роботі із зовнішніми інтерфейсами системи.

Для тестування використовувався такий артефакт, як тестовий сценарій. Тест-кейс – артефакт, мета якого полягає у виконанні певних дій, потрібних для переконання, що система, що розробляється, має функціональність відповідну до описаних вимог [18].

Висновки до розділу 4

У четвертому розділі було описано технології, що використовувалися для розробки вебзастосунку інтернет-видання. Було описано та зображено, як працює обрана архітектура, а саме трирівнева архітектура.

Для візуального розуміння логічної моделі програми на рівні структури було створено діаграму класів, яка представлена по рівням архітектури. Було зображено які класи містить вебзастосунок та з чого вони складаються й які зв’язки вони мають.

Також було описано розробку бази даних і як вона взаємодіє з рівнем доступу до даних. Було наведено приклади використання операцій CRUD.

У останньому підрозділі проведено тестування методом «чорна скриня» для розроблюваного вебзастосунку. Результатом виконання тест-кейсів стало заключення про задовільну якість розроблюваного продукту.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи бакалавра було розроблено вебзастосунок новинний портал. Інтернет-видання мають переваги у швидкості передачі інформації над телебаченням, пресою та радіо. На сайті можна швидко додати, оновити, змінити чи видалити контент.

У результаті кваліфікаційної роботи бакалавра вдалося вирішити наступні завдання:

1. Визначено структурну схему сайту;
2. Створено графічні елементи інтерфейсу сайту, стилів і елементів навігації;
3. Розроблено програмний код, базу даних і інші елементів сайту;

У результаті було створено:

1) Сторінку адміністратора,
2) головну сторінка з слайдером останніх новин та переліком заголовків статтей та картинкою,
3) сторінку реєстрації та логіну,
4) пошук на сайті,
5) реалізовано функціонал вподобання та коментування статтей,
6) сторінку для виведення статті з різними режимами перегляду, які користувач зможе вибрати для додаткової зручності.

4. Протестовано розроблюваний вебзастосунок.

Розроблений вебзастосунок має переваги, які не було реалізовано в проаналізованих існуючих аналогах, а саме зручна навігація, ненагромаджена інформацією головна сторінка, є можливість висловлення думки читачів та реагування на ту чи іншу статтю.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Кашуба, Г. Українські інтернет-видання: комунікативно-лінгвістичні та правові аспекти. Вісник Львів. ун-ту. 2004. Вип. 25. С. 474-480.

2. Лешко, У. Типологія жанрів інтернет-журналістики: теоретичні аспекти. Вісник Львів. ун-ту. 2018. Вип. 44. С. 246–253.

3. Новітні медіа та комунікаційні технології: комплекс навчальних програм для спеціальностей «журналістика», «видавнича справа та редагування», «реклама та зв'язки з громадськістю» / за заг. ред. В. Е.Шевченко. – Київ : Паливода А.В., 2012. 412 с.

4. Інтернет-журналістика. Жанри в інтернеті: навчальний посібник. Запоріжжя : ЗНТУ, 2017, 12с.

5. Колір у вебдизайні: особливості використання: вебсайт. URL: https://brainlab.com.ua/blog/czvet-v-veb-dizajne-osobennosti-ispolzovaniya-i-osnovnye-trendy-2020#title_1 (дата звернення: 22.03.2022).

6. HEX-кодування кольору : вебсайт. URL: https://cases.media/article/hex-koduvannya-koloru (дата звернення: 29.03.2022).

7. Адаптація медіаконтенту до веб-середовища сайтів та соціальних мереж : вебсайт. URL: https://cyberleninka.ru/article/n/adaptatsiya-mediakontenta-k-veb-srede-saytov-i-sotsialnyh-setey (дата звернення: 05.04.2022).

8. Захист контенту від незаконного копіювання, або захист авторських прав в мережі інтернет : вебсайт. URL: https://legalaid.ua/ua/zakhyst-kontentu-vid-nezakonnoho-kopiyuvannya-abo-zakhyst-avtorskykh-prav-v-merezhi-internet/ (дата звернення: 10.04.2022).

9. Як захистити контент сайту від копіювання : вебсайт. URL: https://cityhost.ua/blog/kak-zaschitit-kontent-sayta-ot-kopirovaniya.html (дата звернення: 15.04.2022).

10. 10 сервісів для створення структури сайту : вебсайт. URL: https://habr.com/ru/post/467625/ (дата звернення: 20.04.2022).

11. Інструменти UX для вебдизайну : вебсайт. URL: https://app.flowmapp.com/projects/ (дата звернення: 28.04.2022).

12. Робота з Dapper в ASP.NET Core : вебсайт. URL: https://metanit.com/sharp/aspnet5/26.1.php (дата звернення: 03.05.2022).

13. .NET : вебсайт. URL: https://ru.wikipedia.org/wiki/.NET (дата звернення: 14.05.2022).

14. Введення в MS SQL Server та T-SQL. Що таке SQL Server та T-SQL : вебсайт. URL: https://metanit.com/sql/sqlserver/1.1.php (дата звернення: 23.05.2022).

15. Методи HTTP-запитів: приклади та можливі проблеми : вебсайт. URL: https://highload.today/metody-http-zaprosov/ (дата звернення: 28.05.2022).

16. Діаграма класів (class diagram) : вебсайт. URL: https://studopedya.ru/1-85034.html (дата звернення: 01.06.2022).

17. Створення об'єктів передачі даних (DTO) : вебсайт. URL: https://docs.microsoft.com/en-us/aspnet/web-api/overview/data/using-web-api-with-entity-framework/part-5 (дата звернення: 03.06.2022).

18. Що таке цілісність тест-кейса або як правильно описати тест-кейс, щоб перевірити функціонал? : вебсайт. URL: https://training.qatestlab.com/blog/course-materials/test-case-description/ (дата звернення: 10.06.2022).

ДОДАТОК А

Код програми

NEWSSITE.WEB LEVEL

Startup.cs

```
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using NewsSite.Business.Managers;
using NewsSite.Data.Repositories;
using NewsSite.Model.Configuration;

namespace NewsSite.Web
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        public void ConfigureServices(IServiceCollection services)
        {
            services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
                    .AddCookie(options =>
                    {
                        options.LoginPath = "/login";
                        options.Cookie.Name = "my_app_auth_cookie";
                    });
            services.AddScoped<ArticleManager>();
            services.AddScoped<CommentManager>();
            services.AddScoped<LikeManager>();
            services.AddScoped<UserManager>();
            services.AddScoped<WorkLoggerManager>();

            services.AddScoped<ArticleRepository>();
            services.AddScoped<CommentRepository>();
            services.AddScoped<LikeRepository>();
            services.AddScoped<UserRepository>();
            services.AddScoped<WorkLoggerRepository>();

services.Configure<ApplicationConfiguration>(Configuration.GetSection("ConfigurationSettings
"));
            services.AddControllersWithViews();
        }

        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
            else
            {
```

```
                app.UseExceptionHandler("/Home/Error");
                app.UseHsts();
            }
            app.UseHttpsRedirection();
            app.UseStaticFiles();
            app.UseAuthentication();
            app.UseRouting();
            app.UseAuthentication();
            app.UseAuthorization();
            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllerRoute(
                    name: "default",
                    pattern: "{controller=Presentation}/{action=Index}/{id?}");
            });
        }
    }
}
```

Program.cs

```
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace NewsSite.Web
{
    public class Program
    {
        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }
        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                });
    }
}
```

Requests

AddArticleRequest.cs

```
namespace NewsSite.Web.WebObjects.Requests
{
    public class AddArticleRequest
    {
        public string Title { get; set; }
        public string Image { get; set; }
        public string Text { get; set; }
    }
}
```

AddCommentRequest.cs

```
namespace NewsSite.Web.WebObjects.Requests
{
    public class AddCommentRequest
    {
        public int ArticleId { get; set; }
        public string Message { get; set; }
        public int UserId { get; set; }
    }
}
```

AddLikeRequest.cs

```
namespace NewsSite.Web.WebObjects.Requests
{
    public class AddLikeRequest
    {
        public int UserId { get; set; }
        public int ArticleId { get; set; }
    }
}
```

AddUserRequest.cs

```
namespace NewsSite.Web.WebObjects.Requests
{
    public class AddUserRequest
    {
        public string Name { get; set; }
        public string Email { get; set; }
        public string Password { get; set; }
    }
}
```

DeleteCommentRequest.cs

```
namespace NewsSite.Web.WebObjects.Requests
{
    public class DeleteCommentRequest
    {
        public int Id { get; set; }
    }
}
```

EditCommentRequest.cs

```
namespace NewsSite.Web.WebObjects.Requests
{
    public class EditCommentRequest
    {
        public int CommentId { get; set; }
        public string Message { get; set; }
    }
}
```

LoginRequest.cs

```
namespace NewsSite.Web.WebObjects.Requests
{
    public class LoginRequest
    {
```

```
        public string Login { get; set; }
        public string Password { get; set; }
    }
}
```

RemoveArticleRequest.cs

```
namespace NewsSite.Web.WebObjects.Requests
{
    public class RemoveArticleRequest
    {
        public int Id { get; set; }
    }
}
```

RemoveLikeRequest.cs

```
namespace NewsSite.Web.WebObjects.Requests
{
    public class RemoveLikeRequest
    {
        public int UserId { get; set; }
        public int ArticleId { get; set; }
    }
}
```

UpdateArticleRequest.cs

```
namespace NewsSite.Web.WebObjects.Requests
{
    public class UpdateArticleRequest
    {
        public int Id { get; set; }
        public string Title { get; set; }
        public string Image { get; set; }
        public string Text { get; set; }
    }
}
```

Responses

GetArticlesResponse.cs

```
using NewsSite.Model.Dtos;
using System.Collections.Generic;

namespace NewsSite.Web.WebObjects.Responses
{
    public class GetArticlesResponse
    {
        public IEnumerable<ArticleDto> Articles { get; set; }
    }
}
```

LoginResponse.cs

```
namespace NewsSite.Web.WebObjects.Responses
{
    public class LoginResponse
    {
        public bool IsValid { get; set; }
        public string ErrorMessage { get; set; }
```

```
        public string Email { get; set; }
        public string Password { get; set; }
    }
}
```

Views

Shared

_layout.cshtml

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta name="viewport" charset="utf-8" content="width=device-width, initial-scale=1,
maximum-scale=1, user-scalable=no">
    <link type="text/css" rel="stylesheet" href="~/css/Bootstrap-5.0.2/bootstrap.css">
    <link rel="stylesheet" href="~/css/bootstrap-vue.css">
    <link rel="stylesheet" href="~/css/bootstrap-vue-compatibility-fixes.css">
    <link rel="stylesheet" href="~/css/Site.css">

    <script src="~/js/vue.js"></script>
    <script src="~/js/vue-resource.js"></script>
    <script src="~/js/bootstrap-vue.js"></script>

</head>
<body>
    <div>
        <div class="header fixed-top pb-2" id="layout">
            <div class="row">
                <div class="col-lg-3 col-md-4 col-12">
                    <a href="/Presentation" id="headerLabel">Новини М <img
src="~/images/Ukraine.png" class="flag" /> </a>
                </div>
                <div class="col-lg-8 col-md-6 col-9 pt-1">
                    <div class="input-group searchBlock ">
                        <input class="form-control" placeholder="Пошук..." type="search"
aria-label="Пошук" v-model="searchText">
                        <button class="btn btn-outline-warning text-white " v-
on:click="searchOnClick">Пошук</button>
                    </div>
                </div>
                <div class="col-lg-1 col-md-2 col-3 pt-1">
                    <a v-if="isLogined == false" class="btn btn-outline-warning text-white
login" href="/Account">Вхід</a>
                    <a v-else class="btn btn-outline-warning text-white login" v-
on:click="signOutAccount()">Вийти</a>
                </div>
            </div>
        </div>

        <div class="container mt-4">
            <a href="/"><img src="~/images/logoM.png" class="img-fluid pt-2 logo"
alt="Логотип" /></a>
        </div>
        @RenderBody()
        <div id="empty"></div>
        <footer class="px-5 fixed-bottom">
            <div class="row">
                <div class="col-lg-10 col-md-9 col-8">
                    <a class="arrow" href="#">
                        &#8679;
```

```
                    </a>
                    <p class="d-inline footerP">Всі права захищені Законом України</p>
                </div>
                <div class="col-lg-2 col-md-3 col-4 pt-2">
                    <a href="/Presentation/AboutUs" class="about">
                        Про нас
                        &#128101;
                    </a>
                </div>

            </div>
        </footer>
    </div>
</body>
</html>
<script src="~/js/Presentation/Layout.js"></script>
```

Presentation

Index.cshtml

```
@{
    Layout = "_Layout";
}
<link rel="stylesheet" href="~/css/Presentation/Presentation.css" asp-append-version="true">
<div id="presentation">
    <div class="carousel ">
        <b-carousel id="carousel-1"
                    fade
                    :interval="5000"
                    controls
                    >
            <b-carousel-slide v-for="item in articlesForCarousel"
                              class="carouselImage"
                              :img-src="item.image"
                              >
                <div class="carouselTitle" v-on:click="clickOnArticle(item.id)">
                    {{item.title}}
                </div>
            </b-carousel-slide>
        </b-carousel>
    </div>
    <div class="presentation-content row ">
        <b-card class="col-lg-3 col-md-6 col-xs-12 cardArticle"
                v-on:click="clickOnArticle(item.id)"
                v-for="item in articles"
                :img-src="item.image"
                img-alt="Card Image">
            <label class="cardTitle">{{item.title}}</label>
            <hr>
            <div>
                <label class="heart">&#129505;</label>
            </div>
        </b-card>
    </div>
</div>
<script src="~/js/Presentation/Presentation.js"></script>
```

Article.cshtml

```
@model NewsSite.Model.Dtos.ArticleDto
@{
```

```
    Layout = "_Layout";
}
<link rel="stylesheet" href="~/css/Article/Article.css" asp-append-version="true">
<link rel="stylesheet" href="~/css/AdminTools/AdminTools.css" asp-append-version="true">
<div id="article" class="container">
    <div class="bg-light">
        <div class=" mainContent">
            <div class="topButtons">
                <button v-on:click="changeTemplate(1)" class="btn btn-outline-light btn-sm"
type="button">
                    <img src="~/images\View_1.jpg" class="view" alt="1">
                </button>
                <button v-on:click="changeTemplate(2)" class="btn btn-outline-light btn-sm"
type="button">
                    <img src="~/images\View_2.jpg" class="view" alt="2">
                </button>
                <button v-on:click="changeTemplate(3)" class="btn btn-outline-light btn-sm"
type="button">
                    <img src="~/images\View_3.jpg" class="view" alt="3">
                </button>
            </div>
            <div v-if="templateId==1">
                <div>
                    <p class="articleTitle">{{article.title}}</p>
                    <p class="articleText ">
                        <img :src="article.image" class=" mainImg" alt="Picture">
{{article.text}}
                    </p>
                </div>
                <div class="likePlace">
                    <div class="titleForLike">Вподобати статтю</div>
                    <label class="like" v-on:click="OnLikeBtnClick()"> &#129505;</label>
                    @*<label class="like" v-on:click="OnDeleteBtnClick()">
&#129505;</label>*@
                </div>
                <br />
            </div>
            <div v-if="templateId==2">
                <div>
                    <p class="articleTitle">{{article.title}}</p>
                    <img :src="article.image" class="mainImgTmpl2" alt="Picture">
                    <p class="articleText ">
                        {{article.text}}
                    </p>
                </div>
                <br />
            </div>
            <div v-if="templateId==3">
                <div>
                    <p class="articleTitle">{{article.title}}</p>
                    <p class="articleTextTmpl3 "><img :src="article.image"
class="mainImgTmpl3" alt="Picture">{{article.text}}</p>
                </div>
                <br />

            </div>
        </div>
    </div>
    <div class="mt-5" v-if="currentUserId > 0">
        <div class="row commentField">
            <div class="form-group col-lg-7 col-md-8 col-12">
                <input class="form-control" v-model="comment" placeholder="Введіть
коментар..." type="text" maxlength="200">
                <p class="textError">{{errors}}</p>
```

```
            </div>
            <div class="col-lg-5 col-md-4 col-12 buttonForMinScreen">
                <button v-on:click="sentCommentOnClick()" class="btn btn-outline-warning
textButton" type="submit">Відправити &#10148;</button>
            </div>
        </div>
        <div class="commentsShow bgForComment">
            <div v-for="cmt in comments" class="row border-bottom">
                <div class="userNameShow col-lg-7 col-md-6 col-6">
                    {{cmt.userName}}
                </div>
                <div class="dateShow col-lg-5 col-md-6 col-6">
                    {{cmt.dataCreated}}
                </div>
                <div class="commentShow col-lg-9 col-md-9 col-8">
                    {{cmt.comment}}
                </div>
                <div class=" col-lg-1 col-md-1 col-2" v-if="currentUserId == cmt.userId">
                    <button v-on:click="onBtnEditCommentClick(cmt)" class="btn btn-light"
type="button"> &#128394;</button>
                </div>
                <div class=" col-lg-1 col-md-1 col-2" v-if="currentUserId == cmt.userId">
                    <button v-on:click="onBtnDeleteCommentClick(cmt.id)" class="btn btn-
light" type="button"> &#128465;</button>
                </div>
            </div>
        </div>
    </div>
    <b-modal id="modal-edit-comment" title="Редагувати комент" modal-cancel="OK" dialog-
class="customModal" size="lg">
        <b-container fluid>
            Comment:
            <input class="form-control" type="text" v-model="editCmt.comment">
        </b-container>

        <template #modal-footer="{ ok, cancel }">
            @*<button v-on:click="cancelModalWindow()" class="btn customBtn">
                    Закрити
                </button>*@
            <button v-on:click="onEdit()" class="btn btn-outline-secondary">
                Зберігти
            </button>
        </template>
    </b-modal>
</div>
<script>
    var articleId = @Model.Id;
</script>
<script src="~/js/Article/Article.js"></script>
```

Search.cshtml

```
@using NewsSite.Web.ViewModels;
@model SearchViewModel
@{
    Layout = "_Layout";
}
<link rel="stylesheet" href="~/css/Presentation/Search.css" asp-append-version="true">
<link rel="stylesheet" href="~/css/Presentation/Presentation.css" asp-append-version="true">
<div id="search">
    <div class="container">
        <div v-for="article in searchResult" class="listOfArticles row" v-
on:click="ChooseArticleOnClick(article.id)">
```

```
            <img :src="article.image" class="imgForSearch col-lg-3" />
            <div class="col-lg-9">
                <div class="cardTitle">{{article.title}}</div>
                <div class="cardText">{{article.text}}</div>
            </div>
        </div>

    </div>
</div>
<script type="text/javascript">
    var searchValue = '@Html.Raw(Model.searchText)';
</script>
<script src="~/js/Presentation/Search.js"></script>
```

AddNewArticle.cshtml

```
@model NewsSite.Model.Dtos.ArticleDto
@{
    Layout = "_Layout";
}
<link rel="stylesheet" href="~/css/Article/AddNewArticle.css" asp-append-version="true">
<link rel="stylesheet" href="~/css/AdminTools/RemoveArticle.css" asp-append-version="true">
<link rel="stylesheet" href="~/css/AdminTools/AdminTools.css" asp-append-version="true">
<link rel="stylesheet" href="~/css/Article/Article.css" asp-append-version="true">
<div id="addNewArticle">
    <p class="text-center h4 title">{{articleId==null?'ДОДАТИ СТАТТЮ':'РЕДАГУВАТИ
СТАТТЮ'}}</p>
    <br />
    <div class="container widthForm">
        <div class="form-group">
            <label class="titleForm">Додайте заголовок</label>
            <input type="text" class="form-control" placeholder="Введіть заголовок" v-
model="title">
            <p class="textError">{{errorTitle}}</p>
        </div>
        <div class="form-group">
            <label class="titleForm elementsMargin">Додайте URL картинки</label>
            <input type="text" class="form-control" placeholder="Введіть URL картинки" v-
model="image">
            <p class="textError">{{errorImage}}</p>
        </div>
        <div class="form-group">
            <label class="titleForm elementsMargin">Додайте текст</label>
            <textarea class="form-control" rows="3" placeholder="Введіть текст" v-
model="text"></textarea>
            <p class="textError">{{errorText}}</p>
        </div>
        <div class="elementsMargin buttons">
            <button type="button" class="btn btn-outline-warning w-100 textButton" v-
on:click="addArticleClickBtn">{{articleId==null?'Додати статтю':'Редагувати
статтю'}}</button>
        </div>
        <a href="/Admin/RemoveArticle" class="arrowBack">&larr;</a>
    </div>
</div>
@if (Model != null)
{
    <script>
        var articleId = @Html.Raw(Model.Id);
        var articleTitle = "@Html.Raw(Model.Title)";
        var articleImage = '@Html.Raw(Model.Image)';
        var articleText = "@Html.Raw(Model.Text)";
    </script>
```

```
}
<script src="~/js/Article/AddNewArticle.js"></script>
```

AboutUs.cshtml

```
@{
    Layout = "_Layout";
}
<link rel="stylesheet" href="~/css/Presentation/AboutUs.css" asp-append-version="true">
<div class="container">
    <div class="row general">
        <div class="col-lg-5 col-md-5 col-12 mainText">
            <div class="logoNews">Новини М     &ndash; </div>
            <div class="info">
                це інформаційний портал для тих, хто бажає першим отримувати ексклюзивну
інформацію про події в Україні та світі.
                Хто прагне розуміти, що насправді відбувається і вимагає від ЗМІ
правдивості, неупередженості та поваги до своєї аудиторії.
            </div>
        </div>
        <div class="col-lg-7 col-md-7 col-12">
            <img src="~/images/mainPhotos.png" class="collage" />
        </div>
    </div>
</div>
```

Account

Login.cshtml

```
@{
    Layout = "_Layout";
}
<link rel="stylesheet" href="~/css/Article/AddNewArticle.css" asp-append-version="true">
<link rel="stylesheet" href="~/css/AdminTools/AdminTools.css" asp-append-version="true">
<link rel="stylesheet" href="~/css/Login/Login.css" asp-append-version="true">
<link rel="stylesheet" href="~/css/AdminTools/RemoveArticle.css" asp-append-version="true">
<link rel="stylesheet" href="~/css/Article/Article.css" asp-append-version="true">

<form id="login"
      v-on:submit.prevent="loginUser">
    <div class="container loginForm">
        <p class="textErrorLogIn">{{errorIncorrectLogin}}</p>
        <div class="form-group">
            <label class="titleForm" for="email">Логін</label>
            <input v-model="login" id="email" type="email" placeholder="Введіть email"
class="form-control">
            <p class="textError">{{errorLogin}}</p>
        </div>
        <div class="form-group elementsMargin">
            <label class="titleForm" for="password">Пароль</label>
            <input v-model="password" placeholder="Введіть пароль" id="password"
type="password" class="form-control">
            <p class="textError">{{errorPassword}}</p>
        </div>
        <div class="elementsMargin buttons">
            <button class="btn btn-warning" type="submit">Увійти</button>
            <a href="/Account/Registration" class="btn btn-outline-warning
textButton">Зареєструватися</a>
        </div>
        <a href="/Presentation" class="arrowBack">&larr;</a>
    </div>
```

```
</form>
<script src="~/js/Login/Login.js"></script>
```

Registration.cshtml

```
@{
    Layout = "_Layout";
}

<link rel="stylesheet" href="~/css/Article/AddNewArticle.css" asp-append-version="true">
<link rel="stylesheet" href="~/css/AdminTools/AdminTools.css" asp-append-version="true">
<link rel="stylesheet" href="~/css/Login/Login.css" asp-append-version="true">
<link rel="stylesheet" href="~/css/AdminTools/RemoveArticle.css" asp-append-version="true">
<link rel="stylesheet" href="~/css/Article/Article.css" asp-append-version="true">

<div id="registration">
    <div class="container loginForm">
        <div class="form-group">
            <label class="titleForm" for="name">Ваше ім'я</label>
            <input placeholder="Введіть ім'я" type="text" id="name" class="form-control"
maxlength="20" v-model="name">
            <p class="textError">{{errorName}}</p>
        </div>
        <div class="form-group elementsMargin">
            <label class="titleForm" for="email">Логін</label>
            <input placeholder="Введіть email" type="email" id="email" class="form-control"
v-model="email">
            <p class="textError">{{errorEmail}}</p>
        </div>
        <div class="form-group elementsMargin">
            <label class="titleForm" for="password">Пароль</label>
            <input placeholder="Введіть пароль" id="password" type="password" maxlength="20"
class="form-control" v-model="password">
            <p class="textError">{{errorPassword}}</p>
        </div>
        <div class="form-group elementsMargin">
            <label class="titleForm" for="passwordConfirm">Підтвердження паролю</label>
            <input placeholder="Підтвердіть пароль" id="passwordConfirm" type="password"
maxlength="20" class="form-control"
                   v-model="passwordConfirm">
            <p class="textError">{{errorPasswordConfirm}}</p>
        </div>
        <div class="elementsMargin buttons">
            <button v-on:click="onBtnRegisterUserClick()" type="submit" class="btn btn-
warning">Зареєструватися</button>
        </div>
        <a href="/Account" class="arrowBack">&larr;</a>
    </div>
</div>
<script src="~/js/Login/Registration.js"></script>
```

Admin

AdminTools.cshtml

```
@{
    Layout = "_Layout";
}
<link rel="stylesheet" href="~/css/AdminTools/AdminTools.css" asp-append-version="true">

<div class="container">
```

```
    <p class="text-center h4 title">ІНСТРУМЕНТИ АДМІНІСТРАТОРА</p>
    <br />
    <div class="buttons">
        <a href="/Admin/RemoveArticle" class="btn btn-outline-warning w-75
textButton">Налаштування статті</a>
        <br />
        <br />
        <a href="/Admin/CheckLogs" class="btn btn-outline-warning  w-75 textButton">Журнал
входів</a>
    </div>
</div>
```

RemoveArticle.cshtml

```
@{
    Layout = "_Layout";
}
<link rel="stylesheet" href="~/css/AdminTools/RemoveArticle.css" asp-append-version="true">
<link rel="stylesheet" href="~/css/AdminTools/AdminTools.css" asp-append-version="true">

<div id="adminTool">
    <p class="text-center h4 title">НАЛАШТУВАННЯ СТАТТІ</p>
    <br />
    <div class="container configurePanel">
        <div class="row border bg-light my-2 py-2" v-for="article in articles">
            <div class="col-md-1 col-sm-1 col-1 text-center number">{{article.id}}</div>

            <div class="col-md-7 col-sm-4 col-11  text">{{article.title}}</div>
            <div class="col-md-2 col-sm-2 col-4 text-center">
                <button type="button" class="btn btn-outline-warning"
                        v-on:click="onBtnDeleteClick(article.id)">
                    Видалити
                </button>
            </div>
            <div class="col-md-2 col-sm-2 col-4 text-center">
                <button type="button" class="btn btn-outline-warning"
                        v-on:click="onBtnEditClick(article.id)">
                    Редагувати

                </button>
            </div>
        </div>
        <br />
        <div class="buttons">
            <a href="/Presentation/AddNewArticle" class="btn btn-outline-warning textButton
w-75">Додати статтю</a>
        </div>
        <a href="/Admin/AdminTools" class="arrowBack">
            &larr;
        </a>
    </div>
</div>
<script src="~/js/AdminTools/Remove.js"></script>
```

Controllers

AccountController.cs

```
using Microsoft.AspNetCore.Mvc;
namespace NewsSite.Web.Controllers
{
    public class AccountController : Controller
```

```
    {
        public IActionResult Index()
        {
            return View("Login");
        }
        public IActionResult Registration()
        {
            return View("Registration");
        }
    }
}
```

AdminController.cs

```
using Microsoft.AspNetCore.Mvc;

namespace NewsSite.Web.Controllers
{
    public class AdminController : Controller
    {
        public IActionResult CheckLogs()
        {
            return View();
        }
        public IActionResult AdminTools()
        {
            return View();
        }
        public IActionResult RemoveArticle()
        {
            return View();
        }
    }
}
```

AdminController.cs

```
using Microsoft.AspNetCore.Mvc;
using NewsSite.Business.Managers;
using NewsSite.Model.Enums;
using NewsSite.Web.ViewModels;
using System;

namespace NewsSite.Web.Controllers
{
    public class PresentationController : Controller
    {
        readonly ArticleManager articleManager;
        readonly WorkLoggerManager workLoggerManager;
        public PresentationController(ArticleManager articleManager, WorkLoggerManager
workLoggerManager)
        {
            this.articleManager = articleManager;
            this.workLoggerManager = workLoggerManager;
        }
        public IActionResult Index()
        {
            var remoteIpAddress = HttpContext.Connection.RemoteIpAddress;
            this.workLoggerManager.AddLog(new Model.Dtos.WorkLogDto() { IpAddress =
remoteIpAddress.ToString(), LogEvent = LogEventType.OpenPresentation, LogType =
WorkLogType.Information, Date = DateTime.Now });
            return View();
        }
```

```
        public IActionResult Article(int id)
        {
            var remoteIpAddress = HttpContext.Connection.RemoteIpAddress;
            this.workLoggerManager.AddLog(new Model.Dtos.WorkLogDto() { IpAddress =
remoteIpAddress.ToString(), LogEvent = LogEventType.OpenArticle, LogType =
WorkLogType.Information, Date = DateTime.Now });

            var article = articleManager.GetArticleById(id);
            return View(article);
        }
        public IActionResult AddNewArticle(int id)
        {
            var article = articleManager.GetArticleById(id);
            return View(article);
        }
        public IActionResult AboutUs()
        {
            return View();
        }
        public IActionResult Search(string searchText)
        {
            SearchViewModel searchVModel = new SearchViewModel();
            searchVModel.searchText = searchText;
            return View(searchVModel);
        }
    }
}
```

Api

AccountApiController.cs

```
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using NewsSite.Business.Managers;
using NewsSite.Web.WebObjects.Requests;
using NewsSite.Web.WebObjects.Responses;
using System.Collections.Generic;
using System.Security.Claims;
using System.Threading.Tasks;
namespace NewsSite.Web.Api
{
    [Route("api/account")]
    public class AccountApiController : ControllerBase
    {
        UserManager userManager;
        public AccountApiController(UserManager userManager)
        {
            this.userManager = userManager;
        }
        [Authorize]
        [HttpGet("getlogin")]
        public IActionResult GetLogin()
        {
            return Ok($"Ваш логин: {User.Identity.Name}");
        }
        [HttpPost("login")]
        public async Task<LoginResponse> Login([FromBody] LoginRequest request)
        {
            LoginResponse response = new LoginResponse();
            var user = this.userManager.GetUser(request.Login, request.Password);
```

```
            if(user != null)
            {
                response.IsValid = true;
                response.Email = user.Email;
                response.Password = user.Passwords;
                await Authenticate(user.Email);
            }
            else
            {
                response.IsValid = false;
            }

            return response;
        }
        [HttpPost("signOut")]
        public async Task SignOut([FromBody] LoginRequest request)
        {
            await
HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);
        }
        private async Task Authenticate(string userEmail)
        {
            var claims = new List<Claim>
            {
                new Claim(ClaimsIdentity.DefaultNameClaimType, userEmail)
            };
            ClaimsIdentity id = new ClaimsIdentity(claims, "ApplicationCookie",
ClaimsIdentity.DefaultNameClaimType, ClaimsIdentity.DefaultRoleClaimType);
            await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme,
new ClaimsPrincipal(id));
        }
    }
}
```

AdminController.cs

```
using Microsoft.AspNetCore.Mvc;
using NewsSite.Business.Managers;
using NewsSite.Web.WebObjects.Responses;
namespace NewsSite.Web.Api
{
    [Route("api/admin")]
    public class AdminController : ControllerBase
    {
        WorkLoggerManager workLoggerManager;
        public AdminController(WorkLoggerManager workLoggerManager)
        {
            this.workLoggerManager = workLoggerManager;
        }
        [HttpGet("logs")]
        public GetLogsResponse Logs()
        {
            GetLogsResponse response = new GetLogsResponse();
            response.WorkLogs = this.workLoggerManager.GetLogs();
            return response;
        }
    }
}
```

ArticleController.cs

```
using Microsoft.AspNetCore.Mvc;
using NewsSite.Business.Managers;
```

```
using NewsSite.Model.Responses;
using NewsSite.Web.WebObjects.Requests;

namespace NewsSite.Web.Api
{
    [Route("api/article")]
    public class ArticleController : ControllerBase
    {
        ArticleManager articleManager { get; set; }
        UserManager userManager { get; set; }
        public ArticleController(ArticleManager articleManager, UserManager userManager)
        {
            this.articleManager = articleManager;
            this.userManager = userManager;
        }
        public GetArticleResponse GetArticle(int articleId)
        {
            GetArticleResponse response = new GetArticleResponse();
            response.Article = this.articleManager.GetArticleById(articleId);

            if(User.Identity.Name != null)
            {
                response.CurrentUserId = this.userManager.GetUser(User.Identity.Name).Id;
            }

            return response;
        }
        [HttpPost]
        public void AddArticle([FromBody]AddArticleRequest request)
        {
            this.articleManager.AddArticle(request.Title, request.Text, request.Image);
        }

        [HttpPost("UpdateArticle")]
        public void UpdateArticle([FromBody] UpdateArticleRequest request)
        {
            this.articleManager.UpdateArticle(request.Id, request.Title, request.Image,
request.Text);
        }

        [HttpDelete()]
        public void DeleteArticle([FromBody] RemoveArticleRequest request)
        {
            this.articleManager.DeleteArticle(request.Id);
        }
    }
}
```

CommentController.cs

```
using Microsoft.AspNetCore.Mvc;
using NewsSite.Business.Managers;
using NewsSite.Model.Dtos;
using NewsSite.Web.WebObjects.Requests;
using System.Collections.Generic;

namespace NewsSite.Web.Api
{
    [Route("api/comment")]
    public class CommentController : ControllerBase
    {
        CommentManager commentManager;
        public CommentController(CommentManager commentManager)
        {
```

```
            this.commentManager = commentManager;
        }
        [HttpPost]
        [Route("comment")]
        public void AddComment([FromBody] AddCommentRequest request)
        {
            CommentDto comment = new CommentDto();
            comment.ArticleId = request.ArticleId;
            comment.Comment = request.Message;
            comment.UserId = request.UserId;
            this.commentManager.AddComment(comment);
        }

        [HttpPut]
        [Route("comment")]
        public void EditComment([FromBody] EditCommentRequest request)
        {
            this.commentManager.EditComment(request.CommentId, request.Message);
        }

        [HttpGet]
        [Route("comments")]
        public List<CommentDto> GetCommentsByArticleId(int articleId)
        {
            var comments = this.commentManager.GetComments(articleId);
            return comments;
        }

        [HttpDelete("comment")]
        public void DeleteComment([FromBody] DeleteCommentRequest request)
        {
            this.commentManager.DeleteComment(request.Id);
        }
    }
}
```

LikeApiController.cs

```
using Microsoft.AspNetCore.Mvc;
using NewsSite.Business.Managers;
using System.Collections.Generic;
using NewsSite.Model.Dtos;
using NewsSite.Web.WebObjects.Requests;

namespace NewsSite.Web.Api
{
    [Route("api/like")]
    public class LikeApiController : ControllerBase
    {
        LikeManager likeManager;
        public LikeApiController(LikeManager likeManager)
        {
            this.likeManager = likeManager;
        }

        [HttpGet]
        [Route("like")]
        public List<LikeDto> GetLikesByArticleId(int articleId)
        {
            var likes = this.likeManager.GetLikesByArticleId(articleId);
            return likes;
        }
        [HttpPost]
        [Route("like")]
```

```
        public void AddLike([FromBody] AddLikeRequest request)
        {
            this.likeManager.AddLike(request.UserId, request.ArticleId);
        }

        [HttpDelete()]
        public void DeleteLike([FromBody] RemoveLikeRequest request)
        {
            this.likeManager.DeleteLike(request.UserId,request.ArticleId);
        }
    }
}
```

PresentationApiController.cs

```
using Microsoft.AspNetCore.Mvc;
using NewsSite.Business.Managers;
using NewsSite.Web.WebObjects.Responses;

namespace NewsSite.Web.Api
{
    [Route("api/presentation")]
    public class PresentationApiController : ControllerBase
    {
        ArticleManager articleManager;
        public PresentationApiController(ArticleManager articleManager)
        {
            this.articleManager = articleManager;
        }
        [HttpGet]
        public GetArticlesResponse GetArticles()
        {
            GetArticlesResponse response = new GetArticlesResponse();
            response.Articles = this.articleManager.GetArticles();
            return response;
        }
        [HttpGet]
        [Route("get-articles-search")]
        public GetArticlesResponse GetArticles(string searchText)
        {
            GetArticlesResponse response = new GetArticlesResponse();
            response.Articles = this.articleManager.GetArticles(searchText);
            return response;
        }
    }
}
```

UserController.cs

```
using Microsoft.AspNetCore.Mvc;
using NewsSite.Business.Managers;
using NewsSite.Model.Dtos;
using NewsSite.Web.WebObjects.Requests;
namespace NewsSite.Web.Api
{
    [Route("api/user")]

    public class UserController : ControllerBase
    {
        UserManager userManager;
        public UserController(UserManager userManager)
        {
            this.userManager = userManager;
```

```
        }
        [HttpPost]
        public void AddUser([FromBody] AddUserRequest request)
        {
            UserDto user = new UserDto();
            user.UserName = request.Name;
            user.Email = request.Email;
            user.Passwords = request.Password;
            this.userManager.AddUser(user);
        }
    }

}
```

JS

Layout.js

```
(function () {
    new Vue({
        el: '#layout',
        mounted() {
            if (window.sessionStorage.getItem('email')) {
                this.isLogined = true;
            }
        },
        data: {
            searchText: '',
            isLogined: false
        },
        methods: {
            searchOnClick: function () {
                if (this.searchText.length > 0) {
                    window.location.href = '/Presentation/Search/get-articles-
search?searchText=' + this.searchText;
                }
            },
            signOutAccount: function () {
                Vue.http.post('/api/account/signOut').then(function () {
                    window.sessionStorage.removeItem("email");
                    window.location.href = '/';
                })
            }
        }
    })
})();
```

Presentation.js

```
(function () {
    new Vue({
        el: '#presentation',
        mounted() {
            var self = this;
            Vue.http.get('/api/presentation').then(function (response) {
                self.articles = response.body.articles;
                if (self.articles.length >= 3) {
                    var articleNumber1 = self.articles.length - 1;
                    var articleNumber2 = self.articles.length - 2;
                    var articleNumber3 = self.articles.length - 3;
                    self.articlesForCarousel.push(self.articles[articleNumber1]);
                    self.articlesForCarousel.push(self.articles[articleNumber2]);
```

```
                    self.articlesForCarousel.push(self.articles[articleNumber3]);
                }

            })
        },
        data: {
            articles: null,
            articlesForCarousel: []
        },
        methods: {
            clickOnArticle: function (id) {
                window.location.href = "/Presentation/Article/" + id;
            }
        }
    })
})();
```

Search.js

```
(function () {
    new Vue({
        el: '#search',
        mounted() {
            var self = this;
            Vue.http.get('/api/presentation/get-articles-search?searchText=' +
this.searchText).then(function (response) {
                self.searchResult = response.body.articles;
                console.log(response);
            })
        },
        data: {
            searchResult: [],
            searchText: searchValue,
            error: "",
        },
        methods: {
            ChooseArticleOnClick: function (id) {
                this.error = "";
                window.location.href = "/Presentation/Article/" + id;
            }
        }
    })
})();
```

Login.js

```
(function () {
    new Vue({
        el: '#login',
        mounted() {

        },
        data: {
            errorIncorrectLogin:"",
            errorLogin: "",
            errorPassword: "",
            login: "",
            password: ""
        },
        methods: {
            loginUser: function () {
                this.errorLogin = "";
                this.errorPassword = "";
```

```
                if (!this.validLogin(this.login)) {
                    this.errorLogin = "Вкажіть коректну електронну пошту";
                }

                if (!this.validPassword(this.password)) {
                    this.errorPassword = "Пароль повинен містити принаймні одну цифру та
одну велику та малу літеру та принаймні 8 або більше символів";
                }

                if (this.errorLogin === "" && this.errorPassword === "") {
                    var self = this;
                    Vue.http.post('/api/account/login', { login: this.login, password:
this.password }).then(function (response) {
                        if (response && response.body.isValid === true) {
                            window.sessionStorage.setItem("email", response.body.email);
                            window.location.href = '/';
                        } else if (response && response.body.isValid === false) {
                            self.errorIncorrectLogin = "Такого користувача не існує";
                        }
                    })
                }
            },
            validLogin: function (login) {
                var regex =
/^(([^<>()[\]\\.,;:\s@"]+(\.[^<>()[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-
9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;
                return regex.test(login);
            },
            validPassword: function (password) {
                var reg = /^(?=.*\d)(?=.*[a-z])|(?=.*[а-я])(?=.*[A-Z])|(?=.*[А-Я])[0-9a-zA-
Z]|[0-9а-яА-Я]{8,}$/;
                return reg.test(password);
            }
        }
    })
})();
```

Registration.js

```
(function () {
    new Vue({
        el: '#registration',
        mounted() {

        },
        data: {
            errorName: "",
            errorEmail: "",
            errorPassword: "",
            errorPasswordConfirm: "",

            name: "",
            email: "",
            password: "",
            passwordConfirm: ""
        },
        methods: {
            onBtnRegisterUserClick: function () {
                this.errorName = "";
                this.errorEmail = "";
                this.errorPassword = "";
                this.errorPasswordConfirm = "";
```

```
                if (!this.validName(this.name)) {
                    this.errorName = "Вкажіть ваше ім'я";
                }
                if (!this.validEmail(this.email)) {
                    this.errorEmail = "Вкажіть коректну електронну пошту";
                }

                if (!this.validPassword(this.password)) {
                    this.errorPassword= "Пароль повинен містити принаймні одну цифру та одну
велику та малу літеру та принаймні 8 або більше символів";
                }

                if (this.password != this.passwordConfirm) {
                    this.errorPasswordConfirm = "Пароль повинен співпадати";
                }

                if (this.errorName == "" && this.errorEmail == "" && this.errorPassword ==
"" && this.errorPasswordConfirm == "") {
                    var newUser = {
                        name: this.name,
                        email: this.email,
                        password: this.password
                    }
                    console.log("ObjectUser", newUser);

                    Vue.http.post('/api/user', { Name: newUser.name, Email: newUser.email,
Password: newUser.password }).then(function (response) {
                        alert("Successful registration");
                        window.location.href = '/Account' ;
                    });
                }

            },
            validName: function (name) {
                var regex = /^(?=.*\d)|(?=.*\w)|(?=.*[А-Я]).+$/i;
                return regex.test(name);
            },
            validEmail: function (email) {
                var re =
/^(([^<>()[\]\\.,;:\s@"]+(\.[^<>()[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-
9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;
                return re.test(email);
            },
            validPassword: function (password) {
                var reg = /^(?=.*\d)(?=.*[a-z])|(?=.*[а-я])(?=.*[A-Z])|(?=.*[А-Я])[0-9a-zA-
Z]|[0-9а-яА-Я]{8,}$/;
                return reg.test(password);
            }
        }
    })
})();
```

NEWSSITE.BUSINESS LEVEL

Managers

ArticleManager.cs

```
using NewsSite.Data.Repositories;
using NewsSite.Model.Dtos;
using System.Collections.Generic;

namespace NewsSite.Business.Managers
```

```
{
    public class ArticleManager
    {
        ArticleRepository articleRepository;
        public ArticleManager(ArticleRepository articleRepository)
        {
            this.articleRepository = articleRepository;
        }
        public IEnumerable<ArticleDto> GetArticles()
        {
            var articles = this.articleRepository.GetArticles();
            return articles;
        }
        public IEnumerable<ArticleDto> GetArticles(string searchText)
        {
            var articles = this.articleRepository.GetArticles(searchText);
            return articles;
        }
        public ArticleDto GetArticleById(int id)
        {
            var article = this.articleRepository.GetArticleById(id);
            return article;
        }
        public void AddArticle(string title, string text, string image)
        {
            this.articleRepository.AddArticle(title, text, image);
        }
        public void UpdateArticle(int id, string title, string text, string image)
        {
            this.articleRepository.UpdateArticle(id, title, image, text);
        }
        public void DeleteArticle(int id)
        {
            this.articleRepository.DeleteArticle(id);
        }
    }
}
```

CommentManager.cs

```
using NewsSite.Data.Repositories;
using NewsSite.Model.Dtos;
using System.Collections.Generic;

namespace NewsSite.Business.Managers
{
    public class CommentManager
    {
        CommentRepository commentRepository;
        public CommentManager(CommentRepository commentRepository)
        {
            this.commentRepository = commentRepository;
        }
        public void AddComment(CommentDto comment)
        {
            this.commentRepository.AddComment(comment);
        }
        public List<CommentDto> GetComments(int articleId)
        {
            var comments = this.commentRepository.GetComments(articleId);
            return comments;
        }
        public void DeleteComment(int id)
        {
```

```
            this.commentRepository.DeleteComment(id);
        }
        public void EditComment(int commentId, string message)
        {
            this.commentRepository.EditComment(commentId, message);
        }
    }
}
```

LikeManager.cs

```
using NewsSite.Data.Repositories;
using NewsSite.Model.Dtos;
using System.Collections.Generic;
namespace NewsSite.Business.Managers
{
    public class LikeManager
    {
        LikeRepository likeRepository;
        public LikeManager(LikeRepository likeRepository)
        {
            this.likeRepository = likeRepository;
        }
        public List<LikeDto> GetLikesByArticleId(int articleId)
        {
            var likes = this.likeRepository.GetLikesByArticleId(articleId);
            return likes;
        }
        public void AddLike(int userId, int articleId)
        {
            this.likeRepository.AddLike(userId, articleId);
        }
        public void DeleteLike(int userId, int articleId)
        {
            this.likeRepository.DeleteLike(userId, articleId);
        }
    }
}
```

UserManager.cs

```
using NewsSite.Data.Repositories;
using NewsSite.Model.Dtos;

namespace NewsSite.Business.Managers
{
    public class UserManager
    {
        UserRepository userRepository;
        public UserManager(UserRepository userRepository)
        {
            this.userRepository = userRepository;
        }
        public void AddUser(UserDto user)
        {
            this.userRepository.AddUser(user);
        }
        public UserDto GetUser(string email, string password)
        {
            var user = this.userRepository.GetUser(email,password);
            return user;
        }
        public UserDto GetUser(string email)
```

```
        {
            var user = this.userRepository.GetUser(email);
            return user;
        }
    }
}
```

NEWSSITE.MODEL LEVEL

Configuration

ApplicationConfiguration.cs

```
namespace NewsSite.Model.Configuration
{
    public class ApplicationConfiguration
    {
        public string ConnectionString { get; set; }
    }
}
```

Dtos

ArticleDto.cs

```
namespace NewsSite.Model.Dtos
{
    public class ArticleDto
    {
        public int Id { get; set; }
        public string Title { get; set; }
        public string Text { get; set; }

        public string Image { get; set; }
    }
}
```

CommentDto.cs

```
using System;

namespace NewsSite.Model.Dtos
{
    public class CommentDto
    {
        public int Id { get; set; }
        public int UserId { get; set; }
        public string UserName { get; set; }
        public int ArticleId { get; set; }
        public DateTime DataCreated { get; set; }
        public string Comment { get; set; }
    }
}
```

LikeDto.cs

```
namespace NewsSite.Model.Dtos
{
    public class LikeDto
    {
        public int Id { get; set; }
        public int UserId { get; set; }
```

```
        public int ArticleId { get; set; }
    }
}
```

UserDto.cs

```
namespace NewsSite.Model.Dtos
{
    public class UserDto
    {
        public int Id { get; set; }
        public string UserName { get; set; }
        public string Email { get; set; }
        public string Passwords { get; set; }
    }
}
```

WorkLogDto.cs

```
using NewsSite.Model.Enums;
using System;

namespace NewsSite.Model.Dtos
{
    public class WorkLogDto
    {
        public int Id { get; set; }
        public WorkLogType LogType { get; set; }
        public LogEventType LogEvent { get; set; }
        public string Message { get; set; }
        public DateTime Date { get; set; }
        public string IpAddress { get; set; }
    }
}
```

NEWSSITE.DATA LEVEL

Repositories

ArticleRepository.cs

```
using Dapper;
using System.Collections.Generic;
using System.Data;
using Microsoft.Data.SqlClient;
using System.Linq;
using NewsSite.Model.Dtos;
using Microsoft.Extensions.Options;
using NewsSite.Model.Configuration;

namespace NewsSite.Data.Repositories
{
    public class ArticleRepository
    {
        string connectionString = null;
        public ArticleRepository(IOptions<ApplicationConfiguration>
applicationConfiguration)
        {
            connectionString = applicationConfiguration.Value.ConnectionString;
        }
        public IEnumerable<ArticleDto> GetArticles()
```

```
        {
            using (IDbConnection db = new SqlConnection(connectionString))
            {
                return db.Query<ArticleDto>("SELECT * FROM Article").ToList();
            }
        }
        public IEnumerable<ArticleDto> GetArticles(string searchText)
        {
            using (IDbConnection db = new SqlConnection(connectionString))
            {
                string sql = "SELECT * FROM Article WHERE title LIKE N'%" + searchText + "%'
";
                return db.Query<ArticleDto>(sql).ToList();
            }
        }
        public ArticleDto GetArticleById(int id)
        {
            using (IDbConnection db = new SqlConnection(connectionString))
            {
                return db.Query<ArticleDto>("SELECT * FROM Article WHERE id = @id", new { id
= id }).FirstOrDefault();
            }
        }
        public void AddArticle(string title, string text, string image)
        {
            using (IDbConnection db = new SqlConnection(connectionString))
            {
                db.Query<ArticleDto>("INSERT INTO Article(Title,Image,Text) values (@title,
@image , @text)", new { title = title, image=image, text = text });
            }
        }
        public void UpdateArticle(int id, string title, string text, string image)
        {
            using (IDbConnection db = new SqlConnection(connectionString))
            {
                db.Query<ArticleDto>("UPDATE Article SET [Title]=@title, [Image]=@image,
[Text]=@text  WHERE [Id] = @id", new { title = title, image = image, text = text, id = id
});
            }
        }
        public void DeleteArticle(int id)
        {
            using (IDbConnection db = new SqlConnection(connectionString))
            {
                db.Query<ArticleDto>("DELETE FROM Article WHERE id = @ID", new { id = id });
            }
        }
    }
}
```

CommentRepository.cs

```
using NewsSite.Model.Dtos;
using System;
using Dapper;
using System.Collections.Generic;
using System.Data;
using Microsoft.Data.SqlClient;
using System.Linq;
using Microsoft.Extensions.Options;
using NewsSite.Model.Configuration;

namespace NewsSite.Data.Repositories
{
```

```
    public class CommentRepository
    {
        string connectionString = null;
        public CommentRepository(IOptions<ApplicationConfiguration>
applicationConfiguration)
        {
            connectionString = applicationConfiguration.Value.ConnectionString;
        }
        public void AddComment(CommentDto comment)
        {
            using (IDbConnection db = new SqlConnection(connectionString))
            {
                db.Query<CommentDto>("INSERT INTO Comment (ArticleId, DataCreated, Comment,
UserId) VALUES (@articleId, @datetime, @message, @userId)", new { articleId =
comment.ArticleId, message = comment.Comment, datetime = DateTime.UtcNow, userId = comment
.UserId});
            }
        }
        public void EditComment(int commentId , string message)
        {
            using (IDbConnection db = new SqlConnection(connectionString))
            {
                db.Query<CommentDto>("Update [Comment] set [Comment] = @message where [Id] =
@commentId", new { commentId = commentId, message = message });
            }
        }
        public List<CommentDto> GetComments(int articleId)
        {
            using (IDbConnection db = new SqlConnection(connectionString))
            {
                return db.Query<CommentDto>("SELECT
cmt.[Id],[UserId],[ArticleId],[DataCreated],[Comment], u.UserName as UserName FROM
[dbo].[Comment] cmt left join Users u on cmt.UserId = u.Id WHERE ArticleId = @atrId", new {
atrId = articleId }).ToList();
            }
        }
        public void DeleteComment(int id)
        {
            using (IDbConnection db = new SqlConnection(connectionString))
            {
                db.Query<CommentDto>("DELETE FROM Comment WHERE id = @Id", new { id = id });
            }
        }
    }
}
```

LikeRepository.cs

```
using Dapper;
using Microsoft.Data.SqlClient;
using Microsoft.Extensions.Options;
using NewsSite.Model.Configuration;
using NewsSite.Model.Dtos;
using System.Collections.Generic;
using System.Data;
using System.Linq;

namespace NewsSite.Data.Repositories
{
    public class LikeRepository
    {
        string connectionString = null;
        public LikeRepository(IOptions<ApplicationConfiguration> applicationConfiguration)
        {
```

```
            connectionString = applicationConfiguration.Value.ConnectionString;
        }
        public List<LikeDto> GetLikesByArticleId(int articleId)
        {
            using (IDbConnection db = new SqlConnection(connectionString))
            {
                return db.Query<LikeDto>("SELECT [Id],[UserId],[ArticleId] FROM [Like] WHERE
[ArticleId]=@atrId", new { atrId = articleId }).ToList();
            }
        }
        public void AddLike(int userId, int articleId)
        {
            using (IDbConnection db = new SqlConnection(connectionString))
            {
                db.Query<LikeDto>("INSERT INTO [Like] (UserId, ArticleId) values (@usId,
@artId)", new { usId = userId, artId = articleId });
            }
        }
        public void DeleteLike(int userId, int articleId)
        {
            using (IDbConnection db = new SqlConnection(connectionString))
            {
                db.Query<LikeDto>("DELETE FROM [Like] WHERE UserId=@usId AND ArticleId =
@artId", new { usId = userId, artId = articleId });
            }
        }
    }
}
```

UserRepository.cs

```
using Dapper;
using Microsoft.Data.SqlClient;
using Microsoft.Extensions.Options;
using NewsSite.Model.Configuration;
using NewsSite.Model.Dtos;
using System.Data;
using System.Linq;

namespace NewsSite.Data.Repositories
{
    public class UserRepository
    {
        string connectionString = null;
        public UserRepository(IOptions<ApplicationConfiguration> applicationConfiguration)
        {
            connectionString = applicationConfiguration.Value.ConnectionString;
        }
        public void AddUser(UserDto user)
        {
            using (IDbConnection db = new SqlConnection(connectionString))
            {
                db.Query<UserDto>("INSERT INTO Users (UserName, Email, Passwords) VALUES
(@name, @email, @password)", new { name = user.UserName, email = user.Email, password =
user.Passwords});
            }
        }
        public UserDto GetUser(string email, string password)
        {
            using (IDbConnection db = new SqlConnection(connectionString))
            {
                return db.Query<UserDto>("SELECT * FROM Users WHERE Email = @email AND
Passwords =@password", new { email = email, password = password }).FirstOrDefault();
            }
```

```
        }
        public UserDto GetUser(string email)
        {
            using (IDbConnection db = new SqlConnection(connectionString))
            {
                return db.Query<UserDto>("SELECT * FROM Users WHERE Email = @email", new {
email = email }).FirstOrDefault();
            }
        }
    }
}
```