ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка освітнього Web-ресурсу про війну в Україні мовою C#»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

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

_____ Антон КОНЄВ
(підпис)

Виконав: здобувач вищої освіти групи ПД-41

Антон КОНЄВ

Керівник: Ірина ЗАМРІЙ
д.т.н, доцент

Рецензент: _____

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Конєву Антону Олександровичу

1. Тема кваліфікаційної роботи: «Розробка освітнього Web-ресурсу про війну в Україні мовою C#»
керівник кваліфікаційної роботи д.т.н, доцент, зав. кафедри ІПЗ Ірина ЗАМРІЙ, затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: наукова-технічна література, пов’язана із розробкою вебдодатків; мова програмування C#; фреймворк ASP.NET; система управління базами даних SQLite.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Проаналізувати аналоги інформаційних ресурсів про війну в Україні.
2. Підбір офіційних джерел для збору даних.
3. Розробити інтерфейс користувача для навігації та отримання інформації.
4. Розробити власний парсер для отримання інформації про події в Україні з інформаційних ресурсів.
5. Проаналізувати застосунки з освітніми тестами про війну в Україні.
6. Розробити освітні тести.

7. Спроектувати та розробити веб-застосунок для доступу до освітньої інформації про війну в Україні.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Діаграма варіантів використання.
5. Блок-схема роботи парсеру.
6. Діаграма діяльності.
7. Діаграма класів.
8. Мапа сайту.
9. Екранні форми.
10. Апробація результатів дослідження.

6. Дата видачі завдання «28» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	28.02.-06.03.2024	
2	Аналіз та дослідження існуючих аналогів	07.03-13.03.2024	
3	Підбір офіційних джерел для збору даних.	14.03-17.03.2024	
4	Розробити інтерфейс користувача для навігації та отримання інформації.	18.03-21.03.2024	
5	Проаналізувати застосунки з освітніми тестами про війну в Україні.	22.03-23.03.2024	
6	Розробити власний парсер для отримання інформації про події в Україні з інформаційних ресурсів.	24.03-26.03.2024	
7	Розробити освітні тести.	27.03-29.03.2024	
8	Провести тестування застосунку.	30.03-01.04.2024	
9	Спроектувати та розробити веб-застосунок для доступу до освітньої інформації про війну в Україні.	02.04-28.04.2024	
10	Оформлення роботи: вступ, висновки реферат	29.04-05.05.2024	
11	Розробка демонстраційних матеріалів	06.05-12.05.2024	
12	Попередній захист роботи	13.05-31.05.2024	

Здобувач вищої освіти _____ Антон КОНЄВ
(підпис)

Керівник
кваліфікаційної роботи _____ Ірина ЗАМРІЙ
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 86 стор., 1 табл.,24 рис., 10 джерел.

Мета роботи – створення освітніх матеріалів для шкіл, університетів і самостійного навчання, що допоможуть глибше зрозуміти історичні та соціальні аспекти війни, сприяючи формуванню критичного мислення та громадянської свідомості, а також забезпечення доступу до структурованої інформації про події, пов'язані з війною в Україні, для широкого кола користувачів.

Об’єкт дослідження – процес підвищення обізнаності та самонавчання подіям в Україні.

Предмет дослідження – освітній веб-ресурс для підвищення рівня обізнаності населення про події в Україні.

Короткий зміст роботи: В роботі було реалізовано веб-ресурс, для освіти про війну в Україні з використанням технологій ASP.NET та мови програмування C#. Розроблено алгоритм роботи застосунку та програмно реалізовані ключові функціональні можливості, зокрема: перегляд новин, фото та статистики втрат ворога на фронті; сортування новин; проходження освітніх тестів пов’язаних з війною в Україні. В роботі використано бібліотеку HtmlAgilityPack для парисngу новин та фото, відкритий API “russianpowerRIP” для отримання статистики про втрати ворога на фронті, фреймворк Bootstrap для розробки інтерфейсу, фреймворк ASP.NET для створення веб-додатку на мові C#.

Сферою використання застосунку є освітній сектор, медіа-платформи та громадські організації, спрямовані на підвищення обізнаності та інформованість громадськості щодо поточної ситуації конфлікту в Україні.

КЛЮЧОВІ СЛОВА: ОСВІТА, C#, ASP.NET, ПАРСЕР, НОВИНИ, ТЕСТУВАННЯ.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API - це спосіб завдяки якому дві або більше комп’ютерні програми можуть спілкуватися між собою. Набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення.

MVC - це архітектурна схема, яка складається з трьох компонентів Model, View та Controller, що ефективно відокремлює Business Logic від користувальницького інтерфейсу програми.

URL - це адреса ресурсу в інтернеті. Вона допомагає вашому браузеру знайти певний сайт, сторінку, зображення, файл або відео.

.NET - це безкоштовна кросплатформова платформа розробника з відкритим кодом для створення безлічі програм. Він може запускати програми, написані кількома мовами, при цьому C# є найбільш популярним.

ВСТУП

Зі стрімким розвитком інформаційних технологій з’явилася можливість автоматизувати різноманітні процедури. У сучасному освітньому середовищі, де зростає попит на точну та вичерпну інформацію про конфлікт в Україні, пошук ефективних методів упорядкування та обробки цих даних став актуальним. Отже, створення спеціального веб-ресурсу є необхідним для оптимізації управління даними та забезпечення доступу до ресурсів для освітніх і дослідницьких цілей.

Метою цієї дипломної роботи є створення освітніх матеріалів для шкіл, університетів і самостійного навчання, що допоможуть глибше зрозуміти історичні та соціальні аспекти війни, сприяючи формуванню критичного мислення та громадянської свідомості, а також забезпечення доступу до структурованої інформації про події, пов'язані з війною в Україні, для широкого кола користувачів.

Було виділено такі задачі диплому, як:

- Проаналізувати аналоги інформаційних ресурсів про війну в Україні.
- Підбір офіційних джерел для збору даних.
- Розробити інтерфейс користувача для навігації та отримання інформації.
- Проаналізувати застосунки з освітніми тестами про війну в Україні.
- Розробити власний парсер для отримання інформації про події в Україні з інформаційних ресурсів.
- Розробити освітні тести.
- Спроектувати та розробити веб-застосунок для доступу до освітньої інформації про війну в Україні.

Для створення освітнього веб-сервісу було обрано фреймворк ASP.NET. Він дозволяє легко розробляти веб-застосунки, використовуючи мову програмування C#. Також за допомогою ASP.NET можливо зручно розробляти back-end частини. Для зберігання освітніх тестів було обрано систему управління базами даних SQLite. В додаток до цього було використано мову JS для зберігання даних в Session storage, на стороні браузера(клієнта).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність проекту

Актуальність проекту "Освітній веб-ресурс про війну в Україні" обумовлена низкою важливих факторів, що мають значення як для окремих громадян, так і для суспільства в цілому.

Війна в Україні є однією з найважливіших подій сучасності, яка впливає на мільйони людей. Освітній веб-ресурс допомагає громадянам отримати точну та об'єктивну інформацію про події, що відбуваються, їх причини та наслідки. Багато людей, особливо молодь, можуть не мати достатнього уявлення про історичні, політичні та соціальні аспекти конфлікту, тому цей веб-ресурс може стати важливим освітнім інструментом, що сприяє підвищенню рівня обізнаності та критичного мислення.

У сучасному світі інформаційних війн дуже важливо мати достовірні джерела інформації. Освітній веб-ресурс може допомогти у протидії фейковим новинам та пропаганді, надаючи перевірену та об'єктивну інформацію. Це сприятиме формуванню критичного мислення у користувачів, допомагаючи їм відрізняти факти від маніпуляцій.

Веб-ресурс також може слугувати платформою для збереження свідчень очевидців, документів та інших матеріалів, що мають історичне значення, що важливо для майбутніх поколінь, які будуть вивчати ці події. У сучасному світі інформаційних війн дуже важливо мати достовірні джерела інформації, і освітній веб-ресурс може допомогти у протидії фейковим новинам та пропаганді, надаючи перевірену та об'єктивну інформацію.

Освітні тести цього веб-ресурсу можуть бути інструментом для перевірки знань користувачів щодо подій війни в Україні. Це дозволить їм перевірити своє розуміння та запам'ятовування інформації, яка надається на веб-ресурсі. Такий

підхід сприятиме активнішому залученню аудиторії до матеріалу і забезпечить більш глибоке засвоєння знань.

Крім того, тести можуть бути використані для оцінки рівня освіченості користувачів у певних аспектах війни в Україні. Наприклад, тест може включати питання про історичні факти, ключові події, учасників конфлікту, та інші важливі аспекти. Це дозволить визначити слабкі місця в знаннях і допоможе користувачам зосередитися на конкретних темах для подальшого вивчення.

Таким чином, створення освітнього веб-ресурсу про війну в Україні є надзвичайно актуальним проектом, що відповідає потребам суспільства в умовах сучасних викликів та загроз. Він не лише сприятиме підвищенню рівня обізнаності населення, але й допоможе у формуванні критичного мислення, протидії дезінформації та збереженні історичної пам'яті.

1.2 Аналіз існуючих освітніх веб-ресурсів

Nachasi (https://nachasi.com/) це український веб-сайт, який представляє різноманітні статті, аналітику та матеріали на різноманітні теми, зокрема суспільство, політика, культура, технології та багато іншого. Він надає можливість отримати інформацію про події в Україні та світі через різні формати контенту. На рисунку 1.1 зображено сторінку тесту пов’язаного з війною в Україні.

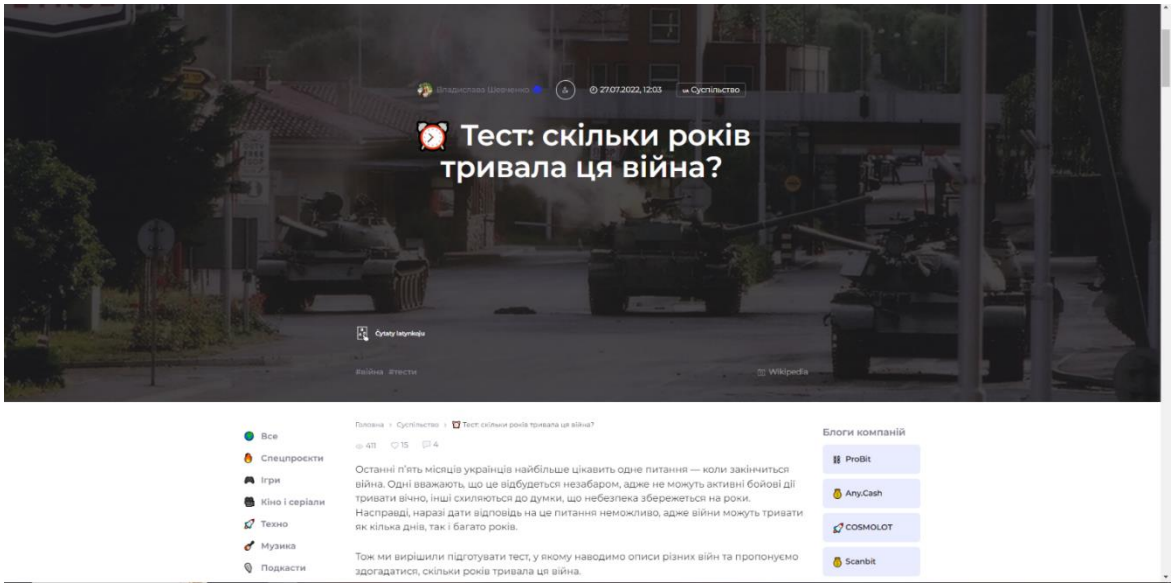


Рис. 1.1 Сторінка тесту пов’язаного з війною в Україні

"Тиждень"(http://www.week.dp.gov.ua/) - це український веб-ресурс, який спеціалізується на публікації освітнього контенту з різних сфер життя. Сайт пропонує широкий спектр матеріалів, таких як статті, аналітика, інтерв'ю, відео та аудіо матеріали на різноманітні теми, включаючи політику, культуру, науку, технології, економіку та інше. Веб-ресурс надає можливість читачам збільшити свої знання та розуміння сучасних подій та тенденцій через доступ до різноманітних освітніх матеріалів. На рисунку 1.2 зображено сторінку з тестом на тему “Революція гідності”.

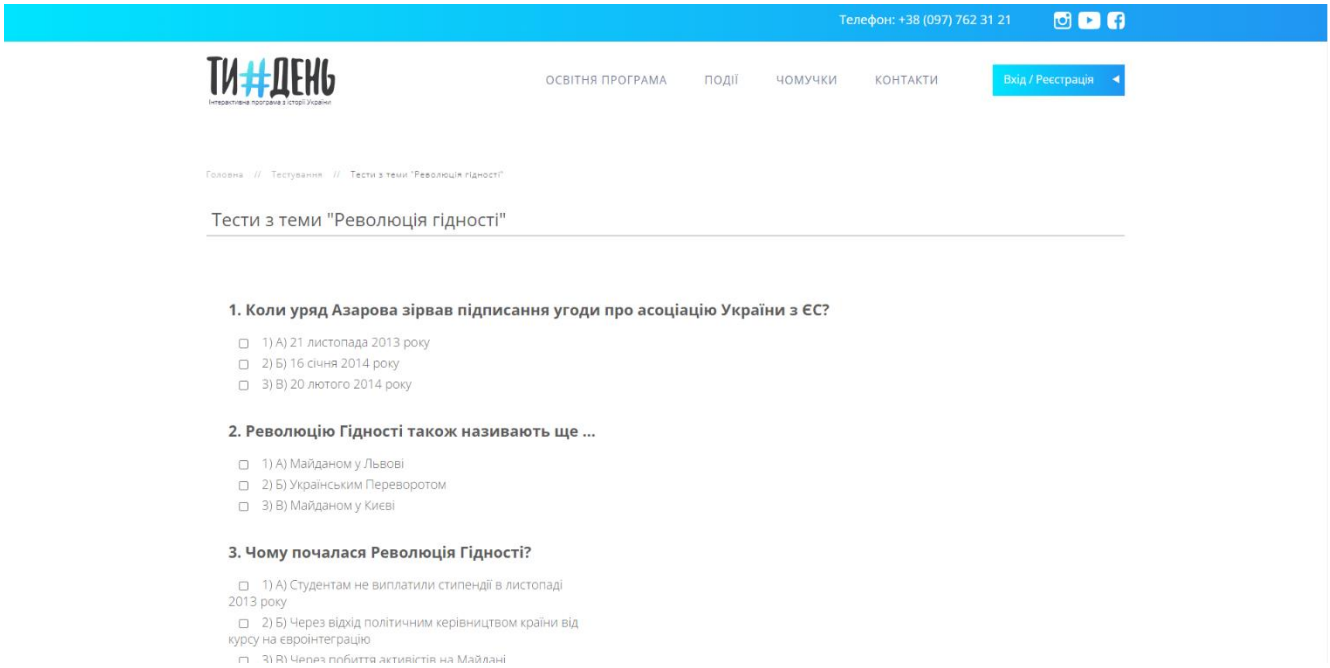


Рис. 1.2 Сторінка з тестом на тему “Революція гідності”

"Накипіло"(https://nakipelo.ua/) - це освітній веб-ресурс, що ставить за мету розширення кругозору та підвищення освітньої грамотності. На сайті зібрана широка колекція цікавих та корисних матеріалів на різноманітні теми: від науки та технологій до культури та мистецтва. "Накипіло" пропонує короткі статті, відео-лекції, аудіо-матеріали та інші формати, які дозволяють з легкістю засвоювати нові знання. На рисунку 1.3 зображено сторінку з тестом на тему “Знаєте, що таке Слобожанщина?”.

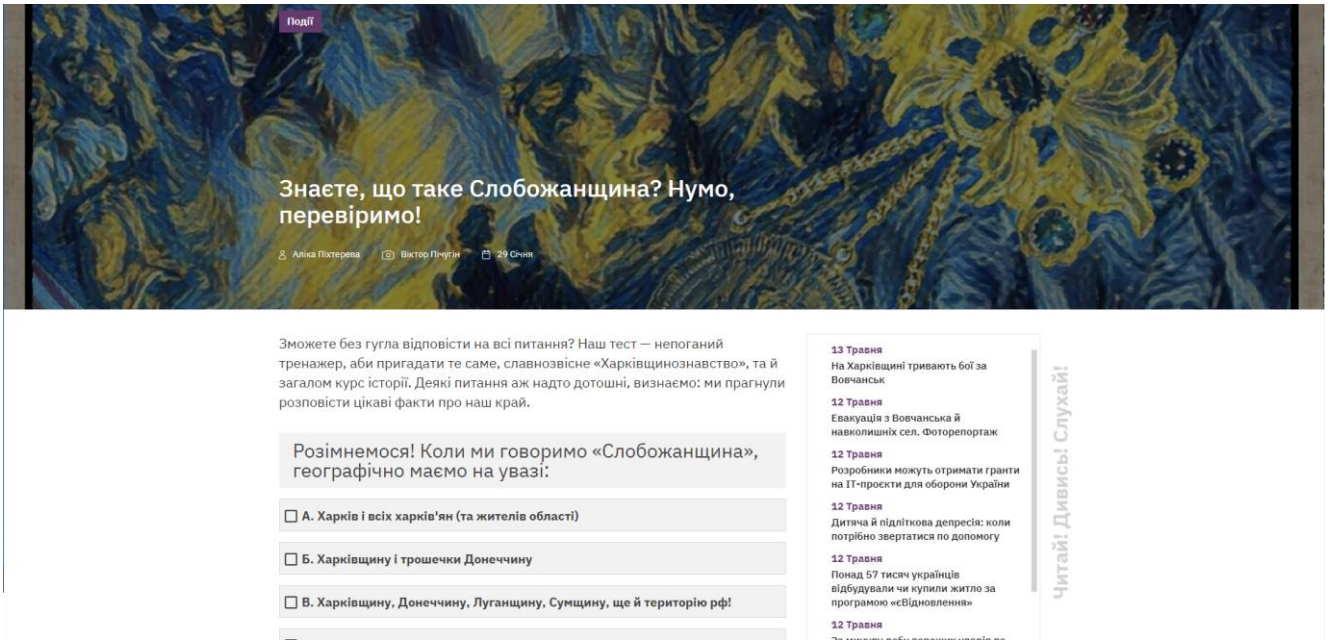


Рис. 1.3 Сторінка з тестом на тему “Знаєте, що таке Слобожанщина?”

На таблиці 1.1 зображена зведена таблиця порівняння аналогів з застосунком за деякими характеристиками.

Таблиця 1.1

Порівняння з існуючих програм-аналогів

	nachcasi.com	week.dp.gov.ua	nakipelo.ua	Ukrainewarinfo
Парсер новин	Ні	Ні	Ні	Так
Доступність через різні пристрої	Так	Так	Так	Так
Наявність сортування новин	Ні	Ні	Ні	Так
Проходження освітніх тестів	Так	Так	Так	Так

1.3 Освітні тести

Освітні тести є невід'ємною складовою частиною будь-якого навчального ресурсу, і освітній веб-ресурс про війну в Україні не є виключенням. Вони виконують кілька важливих функцій, які допомагають користувачам не лише перевірити свої знання, але й глибше зануритися в матеріал, активізуючи процес навчання.

Освітні тести дозволяють користувачам перевірити, наскільки добре вони засвоїли інформацію, представлену на веб-ресурсі. Це особливо важливо для розуміння ключових подій, дат, учасників конфлікту та інших важливих аспектів війни в Україні.

Процес проходження тестів сприяє активнішому залученню користувачів до навчання. Вони стають більш мотивованими до вивчення матеріалу, адже прагнуть отримати високі результати та перевірити свої знання.

Тести можуть містити питання, які спонукають користувачів до роздумів та аналізу, що сприяє розвитку критичного мислення. Наприклад, питання можуть стосуватися причин та наслідків певних подій, аналізу дій різних учасників конфлікту тощо.

Завдяки тестам користувачі можуть виявити, які теми їм потрібно вивчати глибше. Це допомагає сфокусуватися на тих аспектах, які викликають найбільші труднощі, і покращити свої знання.

Тести можуть надавати зворотній зв'язок у вигляді пояснень до правильних та неправильних відповідей. Це допомагає користувачам зрозуміти свої помилки та краще засвоїти матеріал.

Типи освітніх тестів:

- **Множинний вибір**

Це один з найпоширеніших типів тестів, де користувачі повинні вибрати правильну відповідь з декількох варіантів. Такі тести ефективні для перевірки базових знань та фактів.

- **Відкриті питання**

Ці тести передбачають написання коротких або розгорнутих відповідей на запитання. Вони сприяють розвитку навичок критичного мислення та глибшого розуміння матеріалу.

- **Тести на відповідність**

Користувачі повинні зіставити відповідні пари, наприклад, події з датами або особистостей з їхніми діями. Це допомагає структурувати знання та запам'ятовувати зв'язки між різними елементами інформації.

- **Тести на заповнення пропусків**

Користувачам потрібно заповнити пропуски в тексті правильними словами чи фразами. Це сприяє детальному запам'ятовуванню конкретної інформації.

- **Інтерактивні тести**

Включають різні інтерактивні елементи, такі як перетягування відповідей або вибір на зображеннях. Це може зробити процес навчання більш захоплюючим.

Переваги використання освітніх тестів.

- **Підвищення мотивації**

Тести стимулюють інтерес до навчання, оскільки користувачі прагнуть досягти кращих результатів і перевірити свої знання.

- **Адаптація до різних стилів навчання**

Різні типи тестів дозволяють задовольнити потреби користувачів з різними стилями навчання – візуалів, аудіалів, кінестетиків тощо.

- **Оцінка прогресу**

Регулярне тестування дозволяє користувачам відстежувати свій прогрес у вивченні матеріалу та визначати, які теми потребують додаткової уваги.

- **Самоконтроль**

Тести надають можливість самостійно перевіряти свої знання без необхідності зовнішньої оцінки. Це сприяє розвитку автономності в навчанні.

- **Зміцнення пам'яті**

Проходження тестів сприяє активному запам'ятовуванню інформації, оскільки користувачі повторно звертаються до матеріалу та закріплюють свої знання.

Освітні тести на веб-ресурсі про війну в Україні можуть значно підвищити ефективність навчання, допомагаючи користувачам глибше зрозуміти історичні події та їх наслідки, а також сприяти розвитку важливих навичок та компетенцій.

1.4 Висновок до розділу

Підсумовуючи викладене у розділі 1, можна зробити висновок про високу актуальність створення освітнього веб-ресурсу про війну в Україні. Цей проект не лише відповідає нагальним потребам сучасного суспільства у достовірній та об'єктивній інформації, але й має значний потенціал для розвитку критичного мислення та підвищення рівня обізнаності громадян. Війна в Україні є однією з найважливіших подій сучасності, яка суттєво впливає на політичне, соціальне та економічне життя країни, а також на міжнародну спільноту. Тому розробка ресурсу, що надаватиме всебічну інформацію про цей конфлікт, є надзвичайно важливою.

Освітній веб-ресурс сприятиме зменшенню рівня стресу та тривоги серед населення, оскільки доступ до об'єктивної інформації допомагає краще зрозуміти події, що відбуваються. Крім того, такий ресурс може стати платформою для збереження історичної пам'яті, документування свідчень очевидців та інших матеріалів, які мають значну цінність для майбутніх поколінь. У цьому контексті освітній веб-ресурс виступатиме як важливий інструмент протидії дезінформації та пропаганді, що є критично важливим в умовах інформаційних війн.

Освітні тести, інтегровані у веб-ресурс, відіграють ключову роль у навчальному процесі, дозволяючи користувачам перевіряти свої знання, виявляти прогалини у розумінні матеріалу та активно залучатися до навчання. Завдяки різним типам тестів користувачі можуть не лише оцінити рівень своєї обізнаності,

але й зміцнити пам'ять, розвинути навички аналізу та критичного мислення. Тести також забезпечують зворотній зв'язок, що є важливим для самостійного навчання та самоконтролю.

Таким чином, створення освітнього веб-ресурсу про війну в Україні є не просто бажаним, але й необхідним кроком у сучасних умовах. Він надасть громадянам доступ до перевіреної інформації, допоможе підвищити рівень їх обізнаності, сприятиме формуванню критичного мислення та збереженню історичної пам'яті. Важливо також зазначити, що такі ресурси можуть стати потужним інструментом у боротьбі проти дезінформації та фейкових новин, забезпечуючи громадянам надійні джерела знань про ключові події, що визначають наше сьогодення та майбутнє.

2 ПРОЕКТУВАННЯ АРХІТЕКТУРИ ТА ПІДБІР ПРОГРАМНИХ ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ ЗАСТОСУНКУ

2.1 Проектування архітектури застосунку

Застосунок реалізовано з використанням архітектурного паттерна MVC. Він представлений у вигляді діаграми на рисунку 2.1.

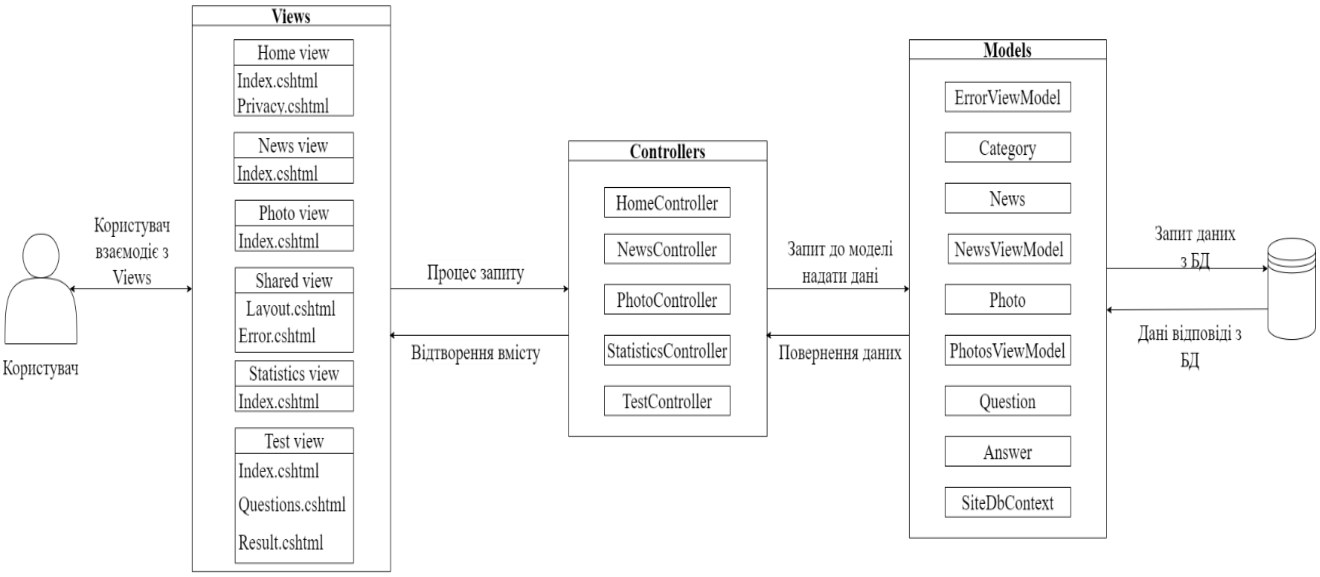


Рис. 2.1 Діаграма архітектури MVC

Користувач взаємодіє з додатком через веб-браузер, відправляючи HTTP-запити і отримуючи HTTP-відповіді. Views відповідають за відображення інформації користувачу. Вони отримують дані від контролерів і форматують їх для відображення. Наприклад, Home view відповідає за відображення головної сторінки з новинами, News view – за сторінку конкретної новини, Photo view – за сторінку фотографій, Shared view – загальне подання макету, яке використовується іншими поданнями для стандартного вигляду сайту, Statistics view – за сторінку статистики втрат ворога, а Test view – за сторінки проходження тестів (Index.cshtml, Questions.cshtml, Result.cshtml).

Контролери обробляють запити користувачів, взаємодіють з моделями для отримання необхідних даних і передають ці дані у Views. HomeController обробляє запити, пов'язані з новинами. NewsController – запити, пов'язані з якоюсь конкретною новиною. PhotoController – запити, пов'язані з фотографіями. StatisticsController – запити, пов'язані зі статистикою втрат ворога. TestController – запити, пов'язані з освітніми тестами, включаючи питання і результати тестів.

Моделі відповідають за бізнес-логіку додатку і взаємодію з базою даних. Наприклад, ErrorViewModel використовується для обробки помилок. Category – для категоризації питань які використовуються у тестах. News – для збереження даних новин. NewsViewModel – для відображення новин. Photo – для збереження даних фотографій. PhotosViewModel – для відображення фотографій. Question – для збереження даних запитань. Answer – для збереження даних відповідей. SiteDbContext – контекст бази даних для управління доступом до даних.

Процес запиту виглядає наступним чином: користувач відправляє запит через веб-браузер, наприклад, доступ до сторінки новин. Запит обробляється відповідним контролером, наприклад, HomeController, який взаємодіє з моделями для отримання необхідних даних. Контролер передає отримані дані відповідному View, наприклад, Home view, яке відображає їх у зручному для користувача форматі. View відтворює отримані дані у вигляді HTML-сторінок, які надсилаються назад користувачу у відповідь на його запит.

2.2 Підбір програмних засобів для реалізації застосунку

2.2.1 Мова програмування C#

C# - це сучасна, об'єктно-орієнтована мова програмування, створена компанією Microsoft у рамках платформи .NET. C# поєднує в собі елементи мов C++ та Java, що робить її легкою для освоєння розробниками, які вже знайомі з цими мовами.

Однією з головних переваг C# є її здатність до створення потужних і масштабованих додатків, включаючи десктопні, веб-додатки, мобільні додатки, а також ігри. Вона підтримує багато сучасних програмних парадигм, включаючи асинхронне програмування, що дозволяє ефективно обробляти великі обсяги даних та забезпечувати високу продуктивність додатків.

C# має строгий типізований синтаксис, що допомагає уникати багатьох поширених помилок, і включає вбудовані засоби для управління пам'яттю, такі як автоматичне збирання сміття (garbage collection). Це значно спрощує процес розробки та обслуговування коду, зменшуючи кількість помилок, пов'язаних з неправильним управлінням пам'яттю.

Інтеграція з платформою .NET надає C# доступ до великої бібліотеки класів, яка містить готові до використання компоненти для вирішення широкого спектра завдань, від роботи з файлами та базами даних до побудови графічного інтерфейсу користувача.

Спільнота розробників C# є однією з найбільших та найактивніших у світі, що сприяє швидкому розвитку мови та наявності великої кількості ресурсів для навчання та підтримки. Крім того, Microsoft постійно оновлює та вдосконалює мову, додаючи нові функції та можливості, що робить C# актуальною і затребуваною серед програмістів.

Завдяки своїй гнучкості, надійності та широкому спектру застосування, C# є одним з найкращих виборів для розробників, які прагнуть створювати якісне та ефективне програмне забезпечення.

2.2.2 Фреймворк ASP.NET

ASP.NET – це потужний веб-фреймворк, розроблений компанією Microsoft, який використовується для створення динамічних веб-додатків і сервісів. Він є частиною платформи .NET і підтримує мову програмування C#. Основною перевагою ASP.NET є можливість розробляти складні веб-додатки з високою продуктивністю та надійністю.

Однією з ключових характеристик ASP.NET є його підтримка архітектурного патерну MVC (Model-View-Controller). MVC розділяє додаток на три окремі компоненти: модель, представлення і контролер. Модель відповідає за бізнес-логіку та управління даними, представлення займається відображенням інформації користувачеві, а контролер обробляє вхідні запити, координуючи взаємодію між моделлю та представленням. Це розділення сприяє кращій організації коду, що полегшує його читання, налагодження та подальше обслуговування.

Крім MVC, ASP.NET підтримує також Razor Pages, що дозволяє створювати динамічні веб-сторінки з мінімальною кількістю коду. Razor Pages використовує синтаксис, який дозволяє вбудовувати C# код безпосередньо в HTML, що спрощує процес розробки і робить його більш інтуїтивним для розробників, особливо тих, хто тільки починає працювати з цим фреймворком.

Ще однією значущою особливістю ASP.NET є його висока продуктивність. Це досягається завдяки компіляції коду в машинний код під час виконання, що значно скорочує час завантаження сторінок і покращує загальну швидкість роботи додатків. Фреймворк також забезпечує потужні інструменти для кешування, обробки помилок та управління станом сеансу, що сприяє підвищенню стабільності та ефективності роботи веб-додатків.

Інтеграція з різноманітними базами даних є ще однією важливою перевагою ASP.NET. Фреймворк підтримує роботу з такими популярними базами даних, як SQL Server, MySQL та PostgreSQL. Для роботи з даними ASP.NET використовує ORM (Object-Relational Mapping) інструменти, зокрема Entity Framework, який дозволяє розробникам легко взаємодіяти з базами даних, виконувати запити та маніпулювати даними без написання складного SQL-коду.

Розробка веб-додатків за допомогою ASP.NET також включає використання таких технологій, як SignalR для створення реальних часом веб-додатків, Web API для розробки RESTful сервісів, а також підтримку сучасних стандартів безпеки, що забезпечує захист даних користувачів і захищає додатки від різних видів атак.

Узагальнюючи, ASP.NET пропонує розробникам потужний і гнучкий інструментарій для створення сучасних веб-додатків, що забезпечує високу

продуктивність, надійність та простоту обслуговування. Завдяки підтримці різних архітектурних підходів і інтеграції з різними технологіями, ASP.NET дозволяє реалізувати широкий спектр веб-рішень, від простих сайтів до складних корпоративних систем.

2.2.3 HtmlAgilityPack

HtmlAgilityPack є потужною бібліотекою для мови програмування C#, яка значно спрощує обробку HTML-коду в програмах. Вона дозволяє розбирати HTML-документи, використовуючи LINQ-подібний синтаксис або стандартні DOM-методи, забезпечуючи зручний доступ до елементів, атрибутів та текстового вмісту сторінки. HtmlAgilityPack дозволяє витягувати дані з HTML-структур, такі як тексти, посилання, зображення та інші елементи, що дозволяє ефективно аналізувати та обробляти веб-контент. Ця бібліотека корисна для створення веб-скраперів, аналізу HTML-змісту та автоматизації обробки веб-сторінок у програмах на C#.

Основні можливості HtmlAgilityPack:

- **Розбір HTML-документів**. HtmlAgilityPack дозволяє легко завантажувати та аналізувати HTML-документи. Ви можете використовувати методи Load для завантаження HTML з файлів, рядків або потоків.
- **Підтримка LINQ**. Завдяки LINQ-подібному синтаксису, розбір HTML стає більш інтуїтивним та зручним. Ви можете використовувати запити LINQ для вибору необхідних елементів з HTML-документа.
- **Маніпуляція елементами.** HtmlAgilityPack надає можливість зручно маніпулювати HTML-елементами, дозволяючи змінювати їх атрибути, текстовий вміст, додавати або видаляти елементи.
- **Витягування даних**. Бібліотека дозволяє витягувати текстовий вміст, посилання, зображення та інші дані з HTML-структур, що робить її ідеальним інструментом для веб-скрапінгу.

- **Підтримка XPath**. HtmlAgilityPack підтримує запити XPath, що дозволяє більш гнучко знаходити елементи в HTML-документі.
- **Обробка помилок**. HtmlAgilityPack може обробляти некоректний HTML-код, автоматично виправляючи деякі помилки, що виникають під час розбору документів.

HtmlAgilityPack є незамінним інструментом для розробників, які працюють з HTML-кодом у C#. Вона надає потужні можливості для розбору, аналізу та маніпуляції HTML-документами, дозволяючи створювати складні та ефективні рішення для роботи з веб-контентом.

2.2.4 HTML

HTML (HyperText Markup Language) є основною мовою розмітки для створення веб-сторінок. Вона використовується для структурування вмісту веб-сторінки за допомогою різних елементів, таких як заголовки, абзаци, списки, таблиці, посилання, зображення тощо. HTML дозволяє визначити, як контент повинен відображатися у браузері, застосовуючи теги, які вказують на тип контенту та його роль на сторінці.

HTML був розроблений як проста мова розмітки для обміну документами в мережі Інтернет. Його основною метою є забезпечення можливості інтеграції тексту з мультимедійними елементами, такими як зображення, відео та інтерактивні форми. Завдяки використанню тегів, HTML дозволяє веб-розробникам організовувати та форматувати вміст таким чином, щоб він був зрозумілий як для людей, так і для машин.

Кожен HTML-документ складається з елементів, які описуються за допомогою тегів. Теги вказують браузеру, як саме потрібно відображати той чи інший вміст. Наприклад, тег <h1> визначає заголовок першого рівня, тег <p> — абзац тексту, а тег <a> використовується для створення гіперпосилань. Крім основних тегів, HTML також підтримує атрибути, які надають додаткову

інформацію про елемент, такі як його ідентифікатор, класи, стилі та інші параметри.

Завдяки HTML можна створювати веб-сторінки з різноманітним візуальним оформленням та функціональністю. HTML дозволяє вбудовувати скрипти та стилі, що розширює можливості веб-розробки. Наприклад, за допомогою атрибута style або зовнішніх CSS-файлів можна змінювати вигляд елементів сторінки, а за допомогою вбудованих скриптів на JavaScript можна додавати інтерактивність і динамічні функції.

HTML поєднується з CSS (Cascading Style Sheets) та JavaScript для створення динамічних та привабливих веб-сторінок, які можуть реагувати на взаємодію користувача та пристосовуватися до різних пристроїв та розмірів екранів. CSS дозволяє окремо визначати стильове оформлення елементів сторінки, включаючи кольори, шрифти, розміри і розташування, тоді як JavaScript забезпечує динамічну взаємодію з користувачем, обробляючи події і змінюючи вміст сторінки в режимі реального часу.

HTML також підтримує семантичну розмітку, яка дозволяє краще структурувати вміст веб-сторінки і зробити його більш доступним для пошукових систем та користувачів з обмеженими можливостями. Семантичні теги, такі як <header>, <footer>, <article> та <section>, допомагають визначити різні частини документа і покращують його зрозумілість та навігацію.

Для розробників веб-ресурсів, таких як освітній веб-ресурс про війну в Україні, HTML є основою, яка дозволяє створювати структуровані, доступні та привабливі веб-сторінки. Він забезпечує основу для інтеграції мультимедійних елементів, форм і скриптів, що разом створюють повноцінний та інтерактивний користувацький досвід.

2.2.5 CSS

CSS (Cascading Style Sheets) — це мова стилів, яка використовується для опису вигляду і форматування документа, написаного на HTML або XML. CSS

дозволяє веб-розробникам розділяти структуру та візуальне представлення контенту, що робить код більш чистим і легким для підтримки.

Основна концепція CSS полягає в тому, що стилі можуть застосовуватися до елементів HTML за допомогою селекторів. Селектори визначають, до яких елементів сторінки буде застосований стиль. Наприклад, селектор h1 буде застосовувати стилі до всіх заголовків першого рівня. Крім того, можна використовувати класи і ідентифікатори для більш точного націлювання стилів.

Каскадність, яка лежить в основі CSS, дозволяє стилям з різних джерел об'єднуватися і перекривати один одного. Стилі можуть бути задані безпосередньо в HTML-документі за допомогою атрибута style, або вони можуть бути визначені у внутрішніх або зовнішніх таблицях стилів. Пріоритетність стилів визначається їхнім розташуванням, специфічністю і правилом важливості !important.

Однією з найбільших переваг CSS є можливість створення адаптивних веб-сторінок, які можуть змінювати свій вигляд залежно від розміру екрана або пристрою, на якому вони переглядаються. Це досягається за допомогою медіазапитів, які дозволяють застосовувати різні стилі в залежності від умов перегляду, таких як ширина вікна браузера, орієнтація екрану, роздільна здатність та інші характеристики пристрою.

CSS також підтримує сучасні можливості оформлення, такі як анімації, переходи та трансформації, які дозволяють створювати складні і динамічні візуальні ефекти. Наприклад, за допомогою анімацій можна плавно змінювати кольори, розміри або позиції елементів, роблячи інтерфейси більш інтерактивними і привабливими для користувачів.

Для створення складних макетів використовуються технології, такі як Flexbox і CSS Grid. Flexbox забезпечує простий і ефективний спосіб розташування елементів у контейнері з гнучкими пропорціями, тоді як CSS Grid дозволяє створювати багатовимірні макети з використанням рядів і стовпців. Ці технології значно полегшують створення адаптивних і масштабованих дизайнів, які можуть автоматично підлаштовуватися під різні розміри екранів і змінювати своє розташування в залежності від доступного простору.

CSS також підтримує змінні (CSS Variables), які дозволяють зберігати значення властивостей і повторно використовувати їх у різних частинах стилів. Це сприяє зменшенню дублювання коду і полегшує його підтримку. Змінні можуть бути визначені на глобальному рівні або всередині конкретних селекторів, що забезпечує гнучкість і модульність стилів.

У контексті розробки освітнього веб-ресурсу про війну в Україні CSS дозволяє створювати естетично привабливі та функціональні веб-сторінки, які будуть однаково добре виглядати на будь-яких пристроях. Використання сучасних методів адаптивного дизайну та динамічного оформлення забезпечує зручність і доступність ресурсу для широкої аудиторії користувачів.

2.2.6 SQLite

SQLite — це легка, вбудована система управління базами даних, яка є популярним вибором для багатьох додатків через свою простоту та зручність використання. Вона не вимагає встановлення серверного програмного забезпечення та забезпечує автономну, надійну і продуктивну роботу з базами даних. Використання SQLite має багато переваг, серед яких невеликий розмір, мінімальні вимоги до налаштування та здатність зберігати всю базу даних у одному файлі. Це робить її ідеальним рішенням для додатків, де потрібна швидка і проста робота з базами даних без необхідності налаштування та обслуговування окремого серверу баз даних.

Однією з головних особливостей SQLite є те, що вона зберігає всю базу даних, включаючи таблиці, індекси, та самі дані, у одному файлі на диску. Це забезпечує простоту в управлінні базами даних, а також високу продуктивність за рахунок мінімізації накладних витрат на взаємодію з файловою системою. Такий підхід дозволяє легко переносити бази даних між різними системами і пристроями.

SQLite підтримує більшість стандартних SQL-запитів, включаючи операції створення, читання, оновлення та видалення даних (CRUD). Вона також забезпечує транзакційну цілісність і використовує механізм журналювання для запобігання

втраті даних у разі збою. Завдяки цьому розробники можуть бути впевнені у надійності збереження даних у своїх додатках.

Ще однією важливою перевагою SQLite є її простота інтеграції у різні мови програмування та платформи. Вона підтримується багатьма популярними мовами, такими як C#, Java, Python та інші. Це робить її універсальним інструментом для розробників, які працюють над різними проектами і потребують надійного та простого рішення для зберігання даних.

У випадку з освітнім веб-ресурсом про війну в Україні, вибір SQLite був обумовлений її здатністю забезпечити ефективне управління даними без необхідності складної налаштування і підтримки. Це дозволяє розробникам зосередитися на реалізації основного функціоналу додатку, не витрачаючи багато часу на роботу з базою даних. Зберігання освітніх тестів, новин та іншої інформації у SQLite забезпечує швидкий доступ до даних і високу продуктивність додатку, що є важливим для забезпечення користувацького досвіду на високому рівні.

2.2.7 Visual Studio

Visual Studio — це інтегроване середовище розробки (IDE) від корпорації Microsoft, яке надає повний набір інструментів для створення програмного забезпечення на різні платформи, включаючи Windows, Android, iOS та веб-додатки. Visual Studio забезпечує зручне та інтуїтивно зрозуміле середовище для написання коду, підтримуючи підсвічування синтаксису, автозавершення коду (IntelliSense) та рефакторинг. Ці функції значно полегшують процес розробки та підвищують продуктивність програмістів.

Однією з основних переваг Visual Studio є його потужні засоби для компіляції та відлагодження коду, що дозволяють швидко виявляти та виправляти помилки. Вбудовані інструменти налагодження підтримують встановлення точок зупинки, перегляд змінних та стеку викликів, що робить процес налагодження зручним і ефективним.

Visual Studio інтегрується з різними системами керування версіями, такими як Git та Azure DevOps, що дозволяє командам розробників ефективно працювати над спільними проектами, відслідковувати зміни та керувати випусками програмного забезпечення.

Інтегровані інструменти для створення інтерфейсу користувача дозволяють розробникам швидко створювати та налаштовувати UI-компоненти для своїх додатків. Visual Studio підтримує WPF, Windows Forms, Xamarin та інші технології для розробки UI, що забезпечує гнучкість у виборі засобів для створення користувацького інтерфейсу.

Для забезпечення високої якості та надійності створених програм Visual Studio містить вбудовані інструменти для тестування, включаючи модульні, інтеграційні та автоматизовані тести UI. Це дозволяє розробникам виявляти помилки на ранніх етапах та забезпечувати стабільність програмного забезпечення.

Visual Studio також пропонує засоби для аналізу коду, що допомагають виявляти потенційні проблеми та оптимізувати продуктивність програм. Інструменти аналізу можуть виявляти недоліки у структурі коду, пропонувати оптимізації та покращувати загальну якість коду.

Інструменти для роботи з базами даних, включені у Visual Studio, дозволяють розробникам легко створювати, редагувати та керувати базами даних безпосередньо з середовища розробки. Інтеграція з SQL Server та іншими базами даних забезпечує ефективне управління даними, що є важливим для сучасних додатків.

Visual Studio підтримує широкий спектр розширень та плагінів, які можна завантажувати з Visual Studio Marketplace. Це дозволяє налаштовувати середовище розробки відповідно до потреб конкретного проекту або команди розробників, роблячи його ще більш гнучким та універсальним інструментом для професійних розробників програмного забезпечення.

2.2.8 Entity Framework

Entity Framework (EF) — це об'єктно-реляційний фреймворк (ORM), розроблений компанією Microsoft для платформи .NET. EF значно спрощує взаємодію з реляційними базами даних, дозволяючи розробникам працювати з даними у вигляді об'єктів, використовуючи знайомі об'єктно-орієнтовані парадигми програмування. Це усуває необхідність написання більшості SQL-запитів вручну, забезпечуючи більш зручний та продуктивний підхід до роботи з базами даних.

EF використовує підхід з моделлю спочатку (Model-First) або з кодом спочатку (Code-First). Модель спочатку передбачає створення концептуальної моделі даних за допомогою візуальних інструментів, після чого фреймворк генерує базу даних на основі цієї моделі. Код спочатку дозволяє розробникам визначати класи сутностей безпосередньо у коді, а потім EF створює відповідну схему бази даних на основі цих класів. Цей підхід забезпечує гнучкість у розробці, дозволяючи розробникам вибирати метод, який найкраще відповідає їхнім потребам.

Однією з ключових функцій Entity Framework є автоматичне відстеження змін. Це означає, що фреймворк автоматично відстежує всі зміни, внесені в об'єкти сутностей, і синхронізує ці зміни з базою даних під час виконання операцій збереження. Ця функція знижує ризик втрати даних і забезпечує їхню цілісність.

Entity Framework також пропонує підтримку LINQ (Language Integrated Query), що дозволяє розробникам виконувати запити до баз даних, використовуючи синтаксис мов програмування C# або VB.NET. Це значно підвищує зрозумілість і підтримку коду, оскільки запити стають частиною коду додатку, а не окремими SQL-запитами.

Підтримка різних постачальників баз даних, таких як SQL Server, MySQL, SQLite та PostgreSQL, є ще однією важливою перевагою Entity Framework. Ця гнучкість дозволяє розробникам легко перемикатися між різними системами баз даних без значних змін у коді додатку.

Entity Framework також включає розширені функції, такі як міграції баз даних, які спрощують процес еволюції схеми бази даних разом із оновленнями додатку. Міграції дозволяють автоматично застосовувати зміни до схеми бази даних без втрати існуючих даних, що є важливим для забезпечення безперервного розвитку і вдосконалення додатків.

Для підвищення продуктивності та масштабованості додатків Entity Framework надає механізми кешування та оптимізації продуктивності. Це дозволяє зменшити навантаження на базу даних і прискорити доступ до часто використовуваних даних.

У контексті розробки освітнього веб-ресурсу про війну в Україні використання Entity Framework дозволяє значно спростити роботу з базами даних, забезпечуючи високу продуктивність і гнучкість розробки. Завдяки цьому розробники можуть зосередитися на реалізації основного функціоналу додатку, не витрачаючи багато часу на управління базою даних та написання складних SQL-запитів.

2.2.9 Javascript

JavaScript — це мова програмування, що зазвичай використовується для додавання інтерактивності до веб-сторінок. Вона дозволяє створювати динамічний вміст, контролювати мультимедіа, анімувати зображення та майже все інше. JavaScript є однією з трьох основних технологій веб-розробки, поряд з HTML та CSS.

JavaScript дозволяє змінювати HTML та CSS, що створює можливості для створення інтерактивних та динамічних веб-сторінок. За допомогою JavaScript можна додавати та видаляти елементи HTML, змінювати стилі CSS у реальному часі, реагувати на дії користувачів, такі як кліки та введення тексту. Асинхронні операції стали можливими завдяки використанню AJAX (Asynchronous JavaScript and XML), що дозволяє JavaScript виконувати запити до серверів без

перезавантаження сторінки, роблячи можливим створення швидких та інтерактивних веб-додатків.

JavaScript також дозволяє реагувати на події, такі як натискання кнопок, переміщення миші, натискання клавіш тощо, що забезпечує більш інтерактивну взаємодію користувача з веб-сторінкою. DOM (Document Object Model) — це структура документа, яка представляє HTML у вигляді дерева елементів, і JavaScript дозволяє маніпулювати цими елементами, змінюючи їх вміст, структуру та стилі. Мова також має багато вбудованих об'єктів, таких як Date, Math, Array та інші, що полегшує роботу з даними та виконання різних операцій.

Однією з ключових переваг JavaScript є його багатоплатформність, адже він виконується в браузері, що робить його доступним для будь-якої платформи, яка підтримує веб-браузери, включаючи настільні комп'ютери, мобільні пристрої та планшети.

JavaScript також підтримує безліч фреймворків та бібліотек, які розширюють його можливості, таких як React, Angular, Vue.js, jQuery та інші. Ці інструменти полегшують розробку складних веб-додатків та зменшують обсяг коду, який потрібно писати.

Node.js — це серверна платформа, яка дозволяє виконувати JavaScript на сервері, розширюючи його можливості за межі браузера. Це дозволяє створювати серверні додатки, працювати з базами даних, файлами та іншими ресурсами.

TypeScript — це надбудова над JavaScript, що додає статичну типізацію, допомагаючи знаходити помилки на етапі компіляції та роблячи код більш зрозумілим і підтримуваним. Модульність JavaScript, підтримувана сучасними версіями, дозволяє розбивати код на невеликі, незалежні частини, що сприяє кращій організації коду та його повторному використанню.

JavaScript є невід'ємною частиною сучасної веб-розробки. Його можливості для створення динамічних та інтерактивних веб-додатків роблять його одним з найпопулярніших мов програмування у світі. Завдяки постійному розвитку та підтримці спільноти, JavaScript залишається актуальним та затребуваним інструментом для розробників.

2.2.10 Bootstrap

Bootstrap є потужним фреймворком для розробки веб-інтерфейсів, що широко використовується розробниками по всьому світу. Його створили Марк Отто і Джейкоб Торнтон в рамках роботи в компанії Twitter, і фреймворк був випущений як проєкт з відкритим вихідним кодом у 2011 році. Bootstrap значно спрощує процес створення веб-сторінок, забезпечуючи розробників набором готових до використання компонентів та інструментів, що дозволяють швидко і ефективно створювати привабливі, адаптивні та функціональні веб-дизайни.

Однією з ключових особливостей Bootstrap є його система сіток, яка дозволяє легко створювати адаптивні макети. Система сіток Bootstrap заснована на гнучких контейнерах, рядках і колонках, які автоматично підлаштовуються під різні розміри екранів і пристроїв. Це робить створення адаптивного дизайну інтуїтивно зрозумілим процесом, забезпечуючи користувачам однаково зручний досвід взаємодії незалежно від типу пристрою, яким вони користуються.

Bootstrap включає великий набір компонентів, таких як кнопки, форми, модальні вікна, каруселі та інші елементи інтерфейсу. Ці компоненти легко налаштовуються за допомогою класів CSS, що дозволяє швидко змінювати їхній зовнішній вигляд відповідно до вимог проєкту. Окрім того, Bootstrap підтримує JavaScript плагіни, які додають динамічності та інтерактивності веб-сторінкам, дозволяючи створювати складніші функціональні можливості, як-от випадаючі меню, каруселі зображень та інші інтерактивні елементи.

Фреймворк також підтримує налаштування через змінні SASS, що дозволяє розробникам легко змінювати кольорову схему, розміри елементів та інші стилістичні параметри без необхідності писати багато додаткового коду. Це сприяє створенню унікальних і персоналізованих дизайнів, зберігаючи при цьому переваги використання стандартних компонентів Bootstrap.

Bootstrap є чудовим інструментом для створення консистентного і професійного дизайну завдяки своїм вбудованим стилям та компонентам.

Фреймворк значно знижує витрати часу на розробку, дозволяючи зосередитися на створенні контенту та функціональності, а не на розробці базових стилів та макетів з нуля.

Сьогодні Bootstrap є одним з найбільш популярних фреймворків для розробки веб-інтерфейсів, і його використовують як початківці, так і досвідчені розробники для створення вебсайтів різної складності. Завдяки своїй гнучкості, потужності і простоті у використанні, Bootstrap продовжує залишатися важливим інструментом у арсеналі веб-розробників по всьому світу.

2.3 Визначення загальних вимог для застосунку

При розробці веб-застосунку було сформульовано кілька функціональних та нефункціональних вимог на основі аналізу існуючих аналогів.

Функціональні вимоги

- **Перегляд новин**. Користувачі мають можливість переглядати список новин, що є актуальними на поточний день. Ця функція передбачає відображення новин у зручному для користувача форматі з можливістю швидкого доступу до повного тексту новини.
- **Проходження освітніх тестів**. Користувачі можуть перевіряти свої знання у сфері "Війна в Україні" за допомогою спеціально розроблених тестів. Ці тести спрямовані на підвищення обізнаності та освітнього рівня користувачів щодо подій та історичних фактів, пов'язаних з війною.
- **Перегляд статистики**. Користувачі можуть переглядати актуальну статистику втрат ворожої піхоти та техніки. Це дозволяє отримувати інформацію про поточний стан бойових дій та оцінювати масштаби втрат.
- **Можливість сортування новин**. Користувачі можуть сортувати новини за заголовком та датою. Ця функція забезпечує зручний доступ до потрібної інформації та підвищує ефективність використання веб-застосунку.

Нефункціональні вимоги

- **Сумісність**. Застосунок повинен бути сумісним з різними пристроями та браузерами для забезпечення доступності для широкого кола користувачів. Це включає підтримку як настільних комп'ютерів, так і мобільних пристроїв.

- **Відповідність контенту**. Весь контент повинен відповідати етичним стандартам та не порушувати законодавства України. Це передбачає перевірку достовірності інформації та уникнення публікації матеріалів, що можуть викликати негативні соціальні або політичні наслідки.

Функціональні вимоги можна представити у вигляді Use Case діаграми, де зображено відношення між акторами (користувачами) та прецедентами. Діаграма показує різні способи використання та різні види користувачів веб-застосунку. Вона допомагає візуалізувати взаємодію між користувачами та системою, виявити всі можливі сценарії використання та забезпечити врахування всіх функціональних вимог при розробці.

Use case діаграма зображена на рисунку 2.2.

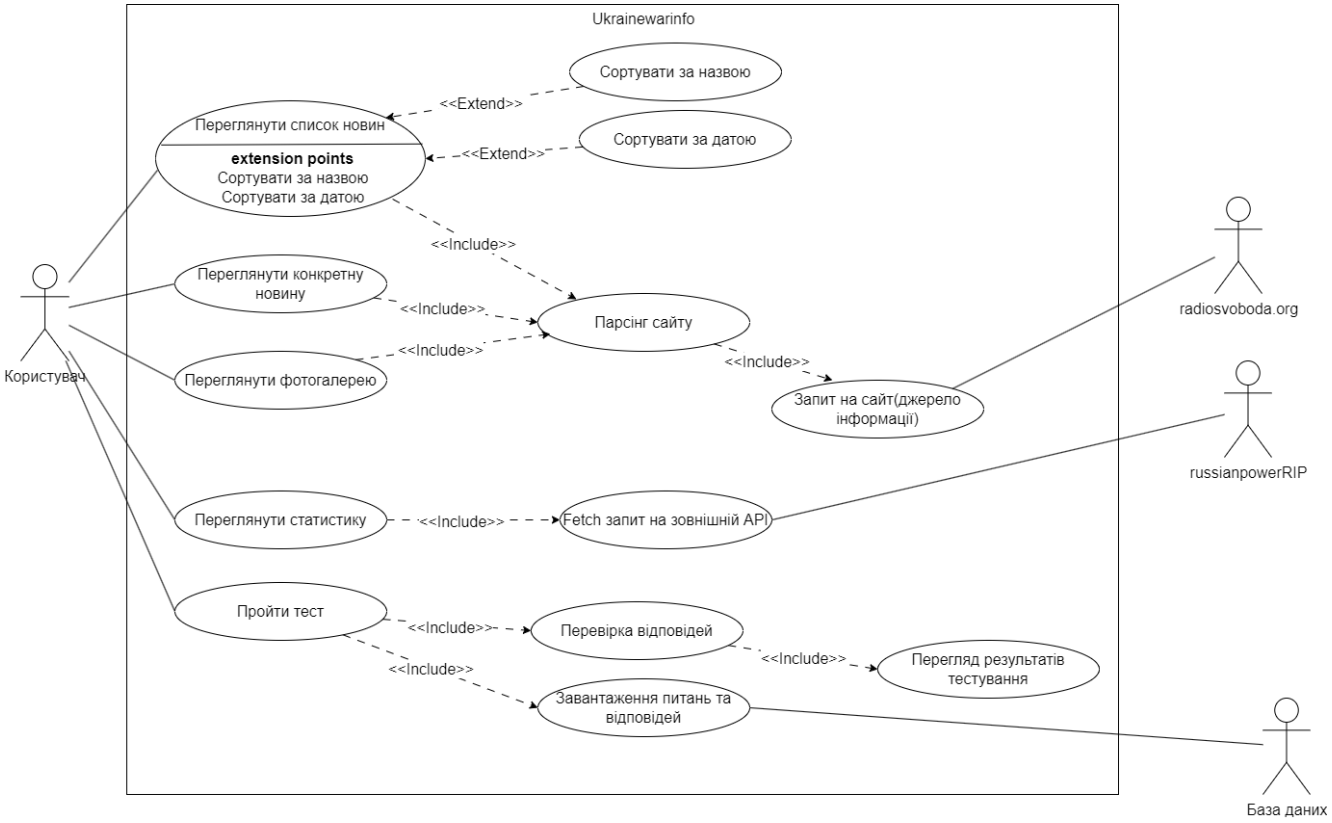


Рис. 2.2 Use case діаграма

3 СТРУКТУРА ТА ПРОЕКТУВАННЯ ЗАСТОСУНКУ

3.1 Діаграма класів

Діаграма класів — це основний засіб для моделювання об'єктно-орієнтованих систем. Вона є графічним представленням класів, інтерфейсів, їх атрибутів, методів і взаємозв'язків між ними. У цьому розділі ми детально розглянемо призначення, структуру і застосування діаграм класів для розробки освітнього веб-застосунку.

Призначення діаграми класів

Діаграма класів служить для декількох ключових цілей:

- **Візуалізація структури системи**. Вона дозволяє розробникам та архітекторам програмного забезпечення бачити, як різні частини системи взаємодіють між собою. Це полегшує розуміння складних систем та їхніх компонентів.
- **Документування системи**. Діаграма класів є важливим документом, який можна використовувати для навчання нових членів команди або для майбутньої підтримки і розвитку системи.
- **Проектування системи**. Використання діаграм класів на етапі проектування дозволяє виявити потенційні проблеми ще до початку написання коду. Це допомагає зекономити час і ресурси, забезпечуючи більш якісний кінцевий продукт.

Структура діаграми класів

Діаграма класів складається з наступних основних компонентів:

- **Класи**. Визначаються основні об'єкти системи. Кожен клас містить атрибути (змінні) та методи (функції), що визначають його поведінку.
- **Асоціації**. Відображають зв'язки між класами. Вони можуть бути односторонніми або двосторонніми, залежно від того, як класи взаємодіють між собою.

- **Агрегації і композиції**. Спеціальні типи асоціацій, що показують, як один клас складається з інших. Агрегація означає "частина-ціле", а композиція — сильніший тип зв'язку, де дочірні елементи існують лише в межах батьківського.
- **Наслідування**. Показує ієрархію класів, де один клас успадковує властивості і методи іншого. Це дозволяє створювати більш гнучкі і повторно використовувані структури.

На рисунку 3.1 зображено діаграму класів для освітнього веб-застосунку про війну в Україні. Вона включає наступні класи:

- **News(Новини)**. Використовується для збереження даних новин, таких як контент новини, дата публікації, ідентифікатор новини, URL зображення, заголовок та URL новини.
- **Photo(Фотографія)**. Використовується для збереження даних фотографій, таких як ідентифікатор, заголовок та джерело зображення.
- **Category(Категорія питання)**. Використовуються для категоризації питань, які присутні в тестах. Містить інформацію про ідентифікатор категорії, назву та питання, які входять до категорії.
- **Answer(Відповідь)**. Використовується для збереження даних відповідей на питання. Містить інформацію про ідентифікатор відповіді, текст відповіді, та ідентифікатор питання до якого вони належать.
- **Question(Питання)**. Використовується для збереження даних запитань, таких як текст, питання, ідентифікатор питання, ідентифікатор правильної відповіді та список відповідей на це питання.

Взаємозв'язки між класами

Зв'язки між класами, що показують, як вони взаємодіють один з одним. Існує кілька типів взаємозв'язків:

- **Асоціація**. Загальний тип зв'язку, що показує, як класи пов'язані один з одним.

- **Агрегація**. Специфічний тип асоціації, що вказує на відношення "ціле-частина", де одна частина може існувати незалежно від цілого.
- **Композиція**. Більш тісний тип агрегації, де частина не може існувати без цілого.
- **Наслідування**. Відношення між базовим класом і похідними класами, що успадковують атрибути і методи базового класу.

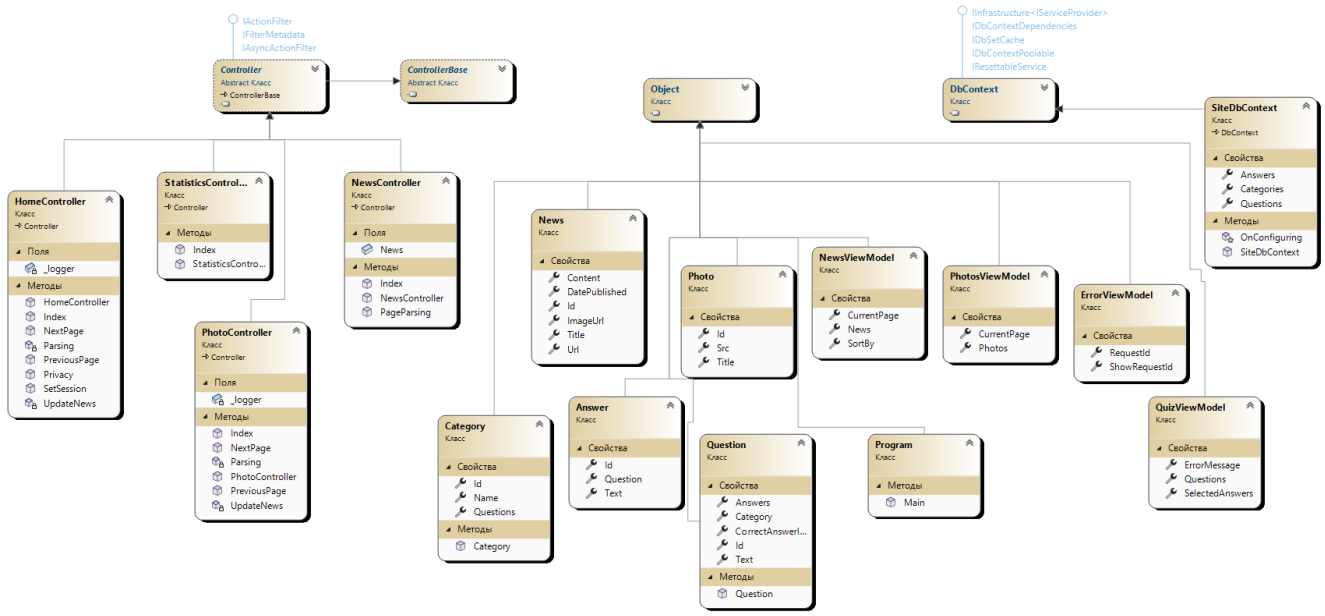


Рис. 3.1 Діаграма класів

3.2 Діаграма діяльності

Діаграма діяльності є важливим інструментом у процесі моделювання програмного забезпечення. Вона використовується для графічного представлення послідовності дій або процесів в системі. Основна мета діаграми діяльності — показати, як різні дії та рішення взаємодіють між собою від початку до завершення певної діяльності.

Основні компоненти діаграми діяльності включають: дії, рішення, потоки, а також початкові та кінцеві стани. Кожна дія представляє конкретний крок або

процес, який виконується системою або користувачем. Точки розгалуження, відомі як рішення, дозволяють процесу йти різними шляхами залежно від певних умов. Потоки з’єднують дії та рішення, вказуючи на напрямок виконання процесу. Початковий стан показує, з чого починається процес, а кінцевий — де він завершується.

Діаграми діяльності широко використовуються для аналізу та проектування програмного забезпечення, моделювання бізнес-процесів, навчання та документування. Вони допомагають розробникам та бізнес-аналітикам зрозуміти порядок виконання дій у системі, виявити можливі проблеми та оптимізувати процеси. Крім того, діаграми діяльності корисні для моделювання бізнес-процесів, надаючи краще розуміння та вдосконалення операцій. У навчанні та документуванні ці діаграми використовуються для створення навчальних матеріалів та документації, пояснюючи, як працює система або процес.

На рисунку 3.2 зображено діаграму діяльності, яка ілюструє процес перегляду новин у системі. Цей процес показує, як користувач взаємодіє з системою для отримання та відображення новин. Ось детальний опис цього процесу:

Користувач розпочинає процес перегляду новин шляхом відправки запиту на сервер. Це може бути ініціалізовано через веб-інтерфейс, наприклад, натисканням на кнопку "Новини" на головній сторінці. Після цього система отримує запит і починає обробляти його.

Перше, що робить система, — це перевірка наявності новин у session storage. Якщо новини доступні, система витягує відповідні дані з нього. Ці дані включають контент новин, дату публікації, заголовки, зображення та URL-адреси новин. У випадку, якщо новини відсутні, система може повернути повідомлення про помилку або спробувати знову пізніше.

Після успішного отримання новин система переходить до етапу обробки даних. На цьому етапі дані можуть бути відфільтровані або відсортовані за певними критеріями, такими як дата публікації або популярність. Це допомагає забезпечити користувача найактуальнішою та найважливішою інформацією.

Далі система форматує дані для відображення. Це включає створення HTML-шаблонів, додавання зображень, заголовків та іншого контенту. Відформатовані дані відправляються назад на клієнтську сторону, де вони відображаються користувачу у вигляді списку новин або окремих статей.

У кінцевому стані користувач може переглядати новини, вибирати окремі статті для детальнішого ознайомлення або повернутися до головної сторінки для вибору іншої категорії новин.

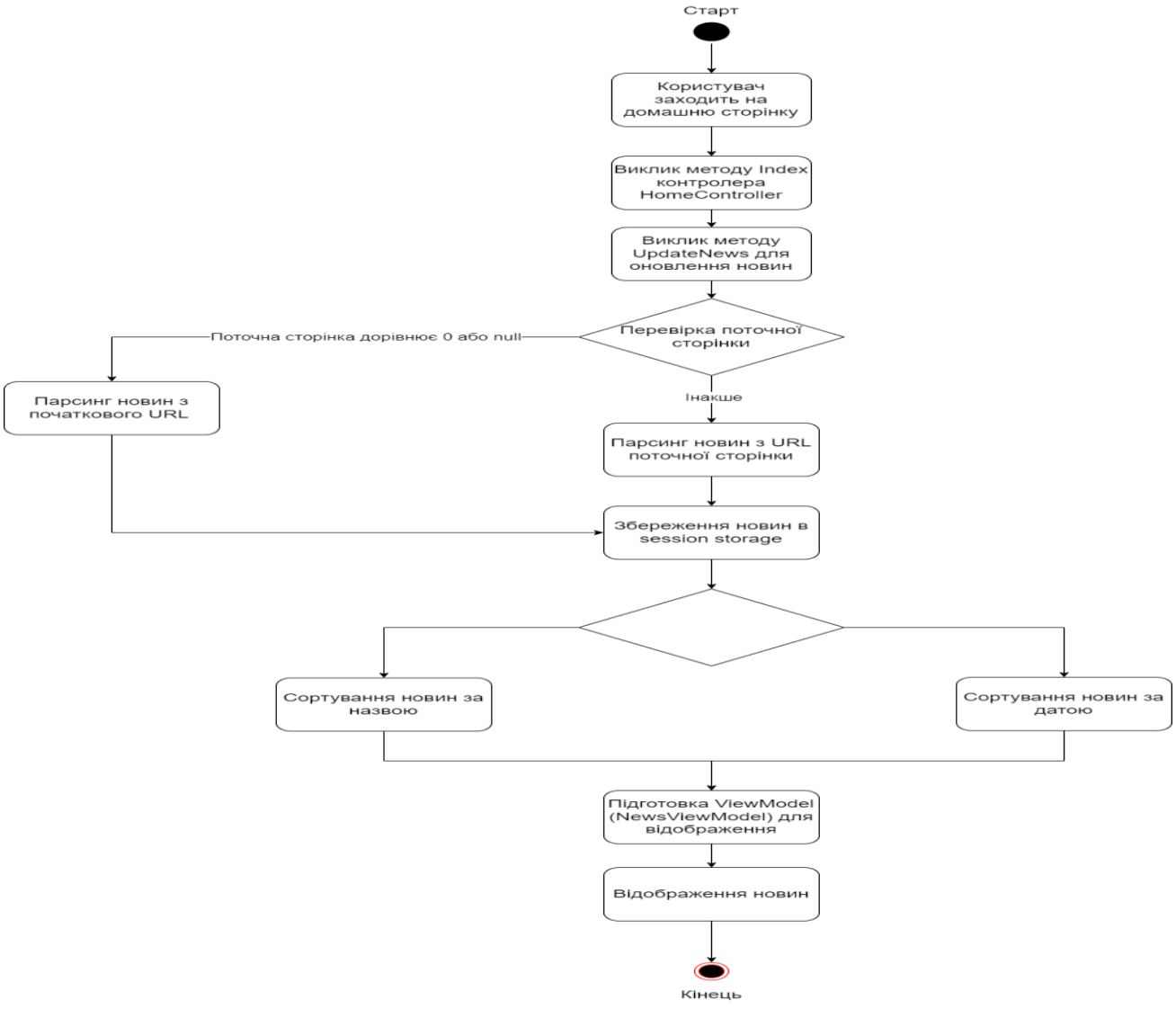


Рис. 3.2 Діаграма діяльності

3.3 Мапа сайту

Мапа сайту являє собою структурну діаграму, яка візуалізує архітектуру веб-сайту шляхом показу всіх доступних сторінок та їх взаємозв'язків. Це інструмент, що допомагає зрозуміти організацію контенту на веб-сайті та навігацію по ньому.

Мапа сайту слугує декільком важливим цілям:

- **Організація контенту**. Мапа сайту відображає структуру веб-сайту, показуючи всі основні сторінки і їх підсторінки. Це допомагає розробникам та дизайнерам зрозуміти, як контент організований та як сторінки взаємодіють між собою.
- **Навігація**. Користувачам, особливо тим, хто вперше відвідує сайт, мапа сайту допомагає швидко знайти необхідну інформацію. Вона показує всі доступні розділи та підрозділи, що полегшує орієнтацію на сайті.
- **SEO (пошукова оптимізація).** Мапи сайту допомагають пошуковим системам, таким як Google, краще індексувати сайт. Чітко структурована мапа сайту може полегшити роботу пошукових алгоритмів, забезпечуючи кращу видимість сайту в результатах пошуку.
- **Ідентифікація проблем**. Аналізуючи мапу сайту, можна виявити потенційні проблеми в організації контенту або навігації. Наприклад, занадто глибокі ієрархії можуть ускладнити доступ до важливої інформації, а відсутність зв'язків між сторінками може створити проблеми для користувачів.

На рисунку 3.3 зображено мапу сайту, яка показує діаграму класів для застосунку. Кожна сторінка представлена у вигляді блоку, а взаємозв'язки між ними — у вигляді стрілок. Це дозволяє відобразити ієрархію сторінок та їх структуру, від основних сторінок до підсторінок.

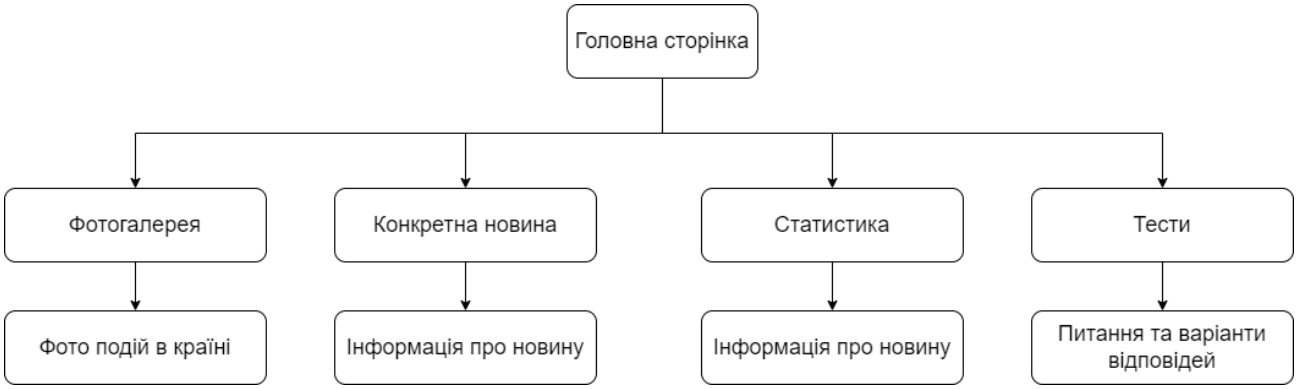


Рис. 3.3 Мапа сайту

3.4 База даних

База даних побудована з використанням Entity Framework та SQLite. Вона містить три основні моделі: Answer, Category та Question, які відображають структуру таблиць та їх взаємозв'язки.

Модель Answer.

Поля:

- Id: Унікальний ідентифікатор відповіді (первинний ключ).
- Text: Текст відповіді.
- Question: Навігаційна властивість, що вказує на пов'язане питання (зовнішній ключ).

На рисунку 3.4 зображено код моделі Answer.

```
namespace Diplom.Models
{
    Ссылок: 3
    public class Answer
    {
        Ссылок: 4
        public int Id { get; set; }
        Ссылок: 3
        public string Text { get; set; } = null!;

        Ссылок: 0
        public virtual Question? Question { get; set; }
    }
}
```

Рис. 3.4 Код моделі Answer

Модель Question.

Поля:

- Id: Унікальний ідентифікатор питання (первинний ключ).
- Text: Текст питання.
- CorrectAnswerIndex: Індекс правильної відповіді серед списку відповідей.
- Answers: Колекція відповідей, пов'язаних з цим питанням.
- Category: Навігаційна властивість, що вказує на категорію, до якої належить це питання (зовнішній ключ).

На рисунку 3.5 зображено код моделі Question.

```
namespace Diplom.Models
{
    Ссылок: 10
    public class Question
    {
        Ссылок: 0
        public Question()
        {
            Answers = new List<Answer>();
        }
        Ссылок: 4
        public int Id { get; set; }
        Ссылок: 2
        public string Text { get; set; } = null!;
        Ссылок: 2
        public int CorrectAnswerIndex { get; set; }

        Ссылок: 7
        public virtual ICollection<Answer> Answers { get; set; }
        Ссылок: 1
        public virtual Category? Category { get; set; }
    }
}
```

Рис. 3.5 Код моделі Question

Модель Category.

Поля:

- Id: Унікальний ідентифікатор категорії (первинний ключ).
- Name: Назва категорії.
- Questions: Колекція питань, що належать до цієї категорії.

На рисунку 3.6 зображено код моделі Category.

```
namespace Diplom.Models
{
    Ссылок: 3
    public class Category
    {
        Ссылок: 0
        public Category()
        {
            Questions = new List<Question>();
        }
        Ссылок: 2
        public int Id { get; set; }
        Ссылок: 1
        public string Name { get; set; } = null!;
        Ссылок: 1
        public virtual ICollection<Question> Questions { get; set; }
    }
}
```

Рис. 3.6 Код моделі Category

Взаємозв'язки між моделями

Question і Answer: Питання має багато відповідей (один-до-багатьох), кожна відповідь належить до одного питання.

Category і Question: Категорія має багато питань (один-до-багатьох), кожне питання належить до однієї категорії.

На рисунку 3.7 зображено діаграму бази даних.

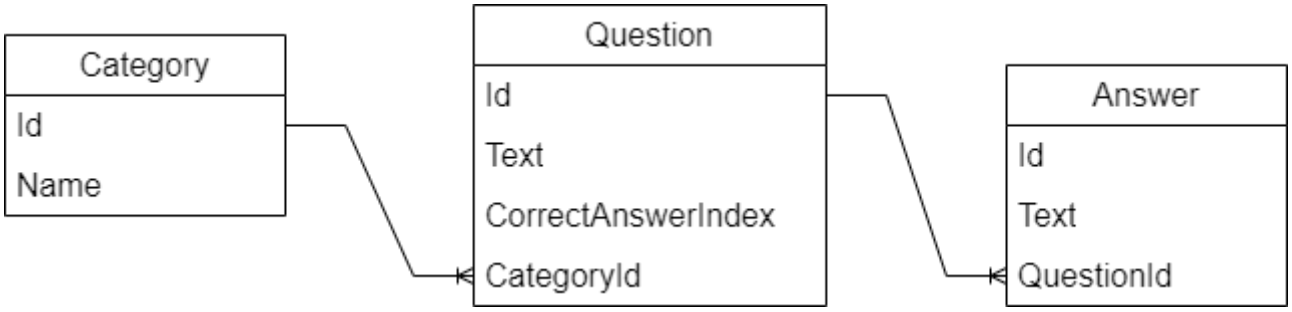


Рис. 3.7 Діаграма бази даних

4 РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ ТА ТЕСТУВАННЯ

4.1 Парсер даних

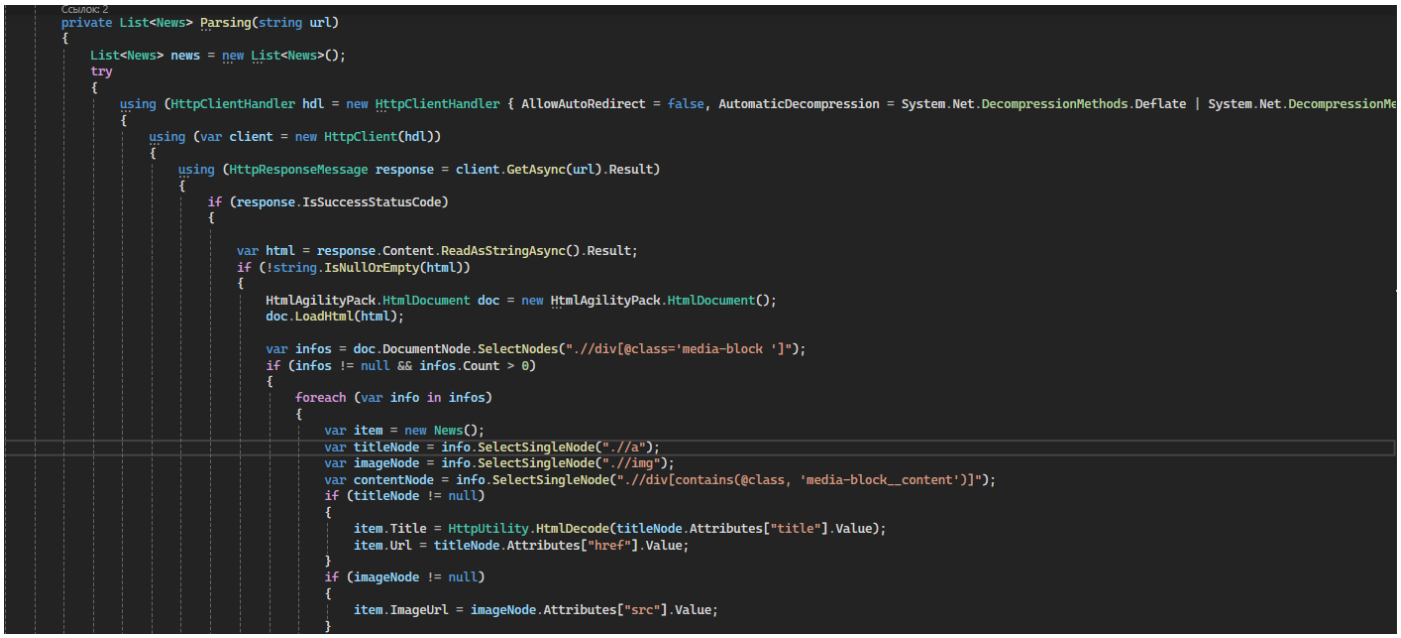
Цей парсер написаний на мові C# та використовує бібліотеку HtmlAgilityPack для розбору HTML-коду веб-сторінки. Його основна мета - отримати інформацію про новини з HTML-коду сторінки, представленої за вказаним URL.

Отже, як цей парсер працює:

- Парсер починається з імпорту необхідних просторів імен та оголошень змінних, таких як список news, який буде заповнений даними про новини, і змінна url, що містить URL-адресу сторінки, яку потрібно спарсити.
- Парсер використовує HttpClient для відправки HTTP-запиту за вказаною URL-адресою та отримання відповіді від сервера.
- Парсер перевіряє, чи успішно завершився запит (якщо код статусу відповіді 200 OK). Якщо запит успішний, він отримує HTML-код сторінки з відповіді.
- Отриманий HTML-код завантажується в HtmlDocument, який надає HtmlAgilityPack, для подальшого аналізу та витягнення даних. Парсер шукає елементи на сторінці, які відповідають заданому селектору .//div[@class='media-block '], який, містить інформацію про новини.
- Для кожного знайденого елемента новини парсер шукає заголовок, URL, URL зображення та дату публікації. Заголовок і URL витягаються з елемента <a>, а URL зображення - з елемента <img>. Дата публікації витягається з елемента, що містить текст дати.
- Парсер обгортає виконання коду в блок try-catch, щоб обробляти будь-які виключення, які можуть виникнути під час виконання, і виводити повідомлення про помилку у разі їх виникнення.
- Після того, як всі новини були витягнуті та додані до списку news, парсер повертає цей список.

Загальний принцип дій полягає в тому, щоб завантажити HTML-контент сторінки, отримати необхідні дані за допомогою HtmlAgilityPack і створити об'єкти новин, використовуючи ці дані.

На рисунках 4.1 та 4.2 зображено код парсеру.

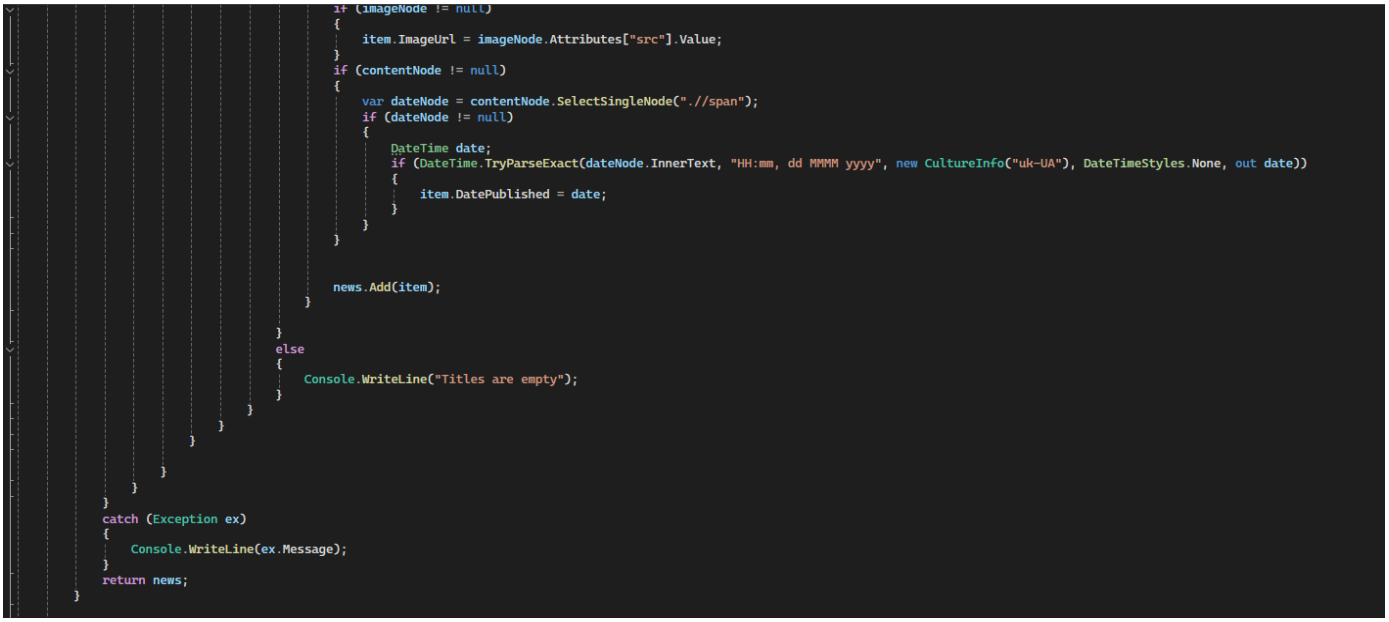


```
Ссылок: 2
private List<News> Parsing(string url)
{
    List<News> news = new List<News>();
    try
    {
        using (HttpClientHandler hdl = new HttpClientHandler { AllowAutoRedirect = false, AutomaticDecompression = System.Net.DecompressionMethods.Deflate | System.Net.DecompressionMe
        {
            using (var client = new HttpClient(hdl))
            {
                using (HttpResponseMessage response = client.GetAsync(url).Result)
                {
                    if (response.IsSuccessStatusCode)
                    {

                        var html = response.Content.ReadAsStringAsync().Result;
                        if (!string.IsNullOrEmpty(html))
                        {
                            HtmlAgilityPack.HtmlDocument doc = new HtmlAgilityPack.HtmlDocument();
                            doc.LoadHtml(html);

                            var infos = doc.DocumentNode.SelectNodes(".//div[@class='media-block ']");
                            if (infos != null && infos.Count > 0)
                            {
                                foreach (var info in infos)
                                {
                                    var item = new News();
                                    var titleNode = info.SelectSingleNode(".//a");
                                    var imageNode = info.SelectSingleNode(".//img");
                                    var contentNode = info.SelectSingleNode(".//div[contains(@class, 'media-block__content')]");
                                    if (titleNode != null)
                                    {
                                        item.Title = HttpUtility.HtmlDecode(titleNode.Attributes["title"].Value);
                                        item.Url = titleNode.Attributes["href"].Value;
                                    }
                                    if (imageNode != null)
                                    {
                                        item.ImageUrl = imageNode.Attributes["src"].Value;
                                    }
```

Рис. 4.1 Код парсеру



```
                                    if (imageNode != null)
                                    {
                                        item.ImageUrl = imageNode.Attributes["src"].Value;
                                    }
                                    if (contentNode != null)
                                    {
                                        var dateNode = contentNode.SelectSingleNode(".//span");
                                        if (dateNode != null)
                                        {
                                            DateTime date;
                                            if (DateTime.TryParseExact(dateNode.InnerText, "HH:mm, dd MMMM yyyy", new CultureInfo("uk-UA"), DateTimeStyles.None, out date))
                                            {
                                                item.DatePublished = date;
                                            }
                                        }
                                    }


                                    news.Add(item);
                                }
                            }
                            else
                            {
                                Console.WriteLine("Titles are empty");
                            }
                        }
                    }

                }
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
        return news;
    }
```

Рис. 4.2 Код парсеру (Продовження)

На рисунку 4.3 зображено блок-схему роботи парсера.

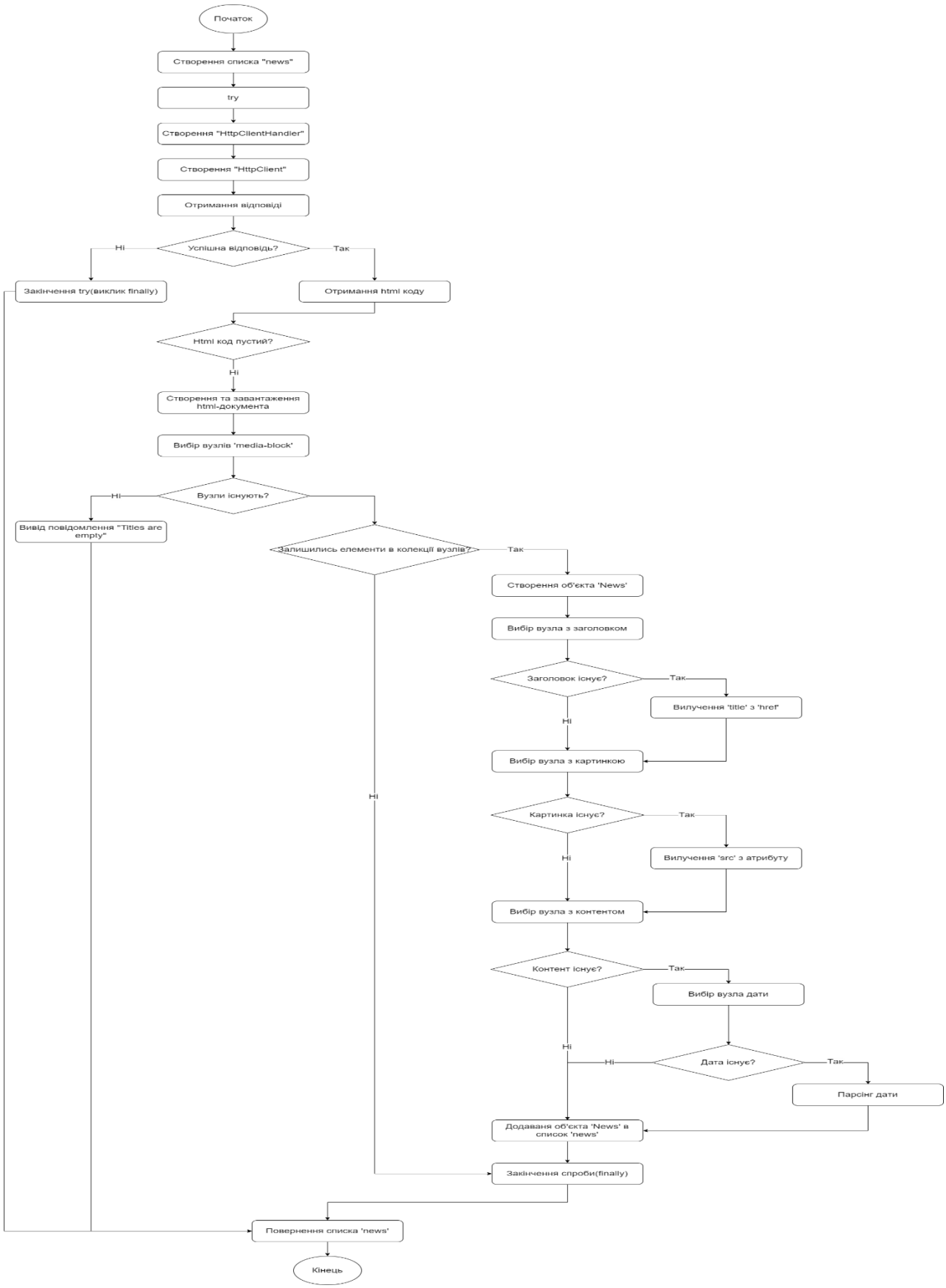


Рис. 4.3 Блок-схема роботи парсера

На рисунку 4.4 зображено інтерфейс сторінки з списком новин.

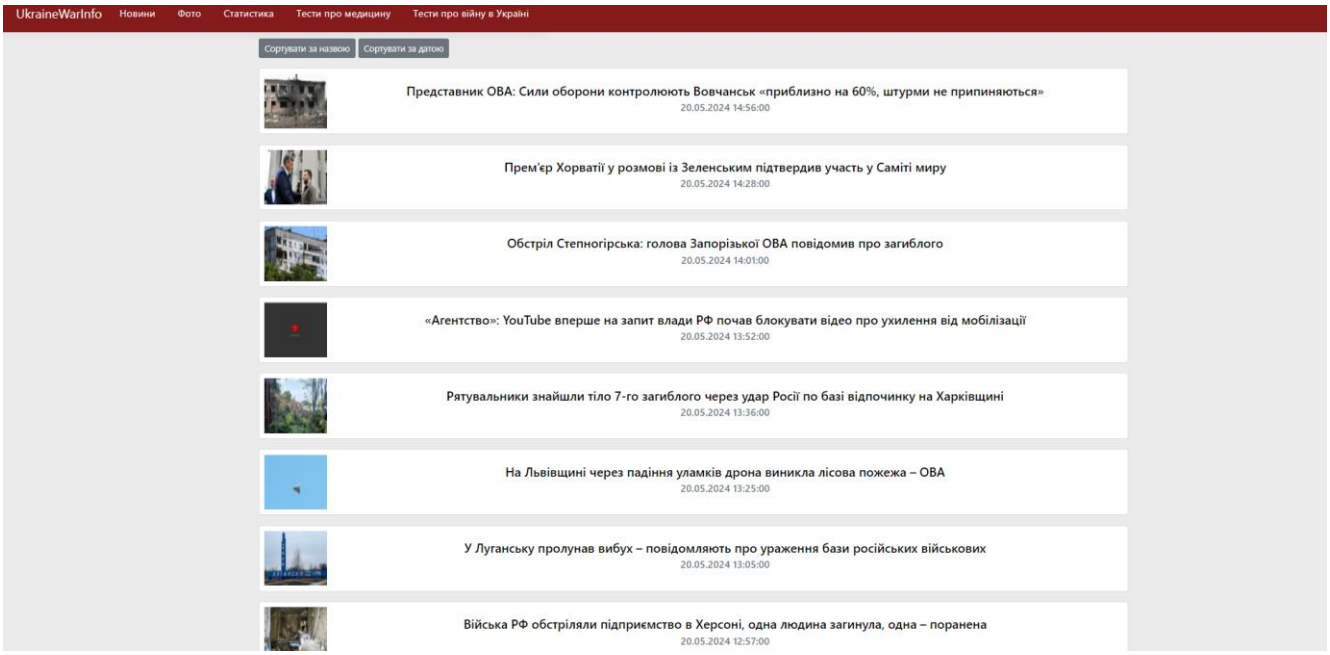


Рис. 4.4 Сторінка зі списком новин

4.2 Скріпт для отримання даних з зовнішнього API

Цей скріпт має функцію getAllInfoAboutWar(), яка виконує запит до веб-сервера, щоб отримати статистику про втрати ворога на фронті. Вона використовує метод fetch(), який є стандартним способом взаємодії з веб-серверами для отримання даних, потім відправляє запит на зовнішній API russianpowerRIP.

Після того, як запит відправлений, ми використовуємо методи ланцюжка .then() та .catch(), щоб обробити результат асинхронного запиту.

Метод .then() викликається, коли ми успішно отримали відповідь від сервера. У нашому випадку, ми очікуємо, що сервер відповість у форматі JSON. Метод .then() обробляє цей JSON-об'єкт, викликаючи функцію _displayItems(data), де data - це отримані дані про військовий стан.

Функція _displayItems(data) призначена для відображення отриманих даних на веб-сторінці. Вона використовує метод document.getElementById() для вибору відповідних HTML-елементів за їх ID, а потім оновлює вміст цих елементів згідно інформацією про військовий стан.

Метод .catch() обробляє помилки, які можуть виникнути під час виконання запиту. У випадку помилки викликається функція console.error(), яка виводить повідомлення про помилку в консоль.

Отже, цей код виконує запит до сервера, отримує дані про статистику втрат ворога на фронті, а потім відображає ці дані на веб-сторінці, забезпечуючи користувачу інформацію про стан військових сил противника.

На рисунку 4.5 зображено інтерфейс сторінки із статистикою втрат ворога на фронті.

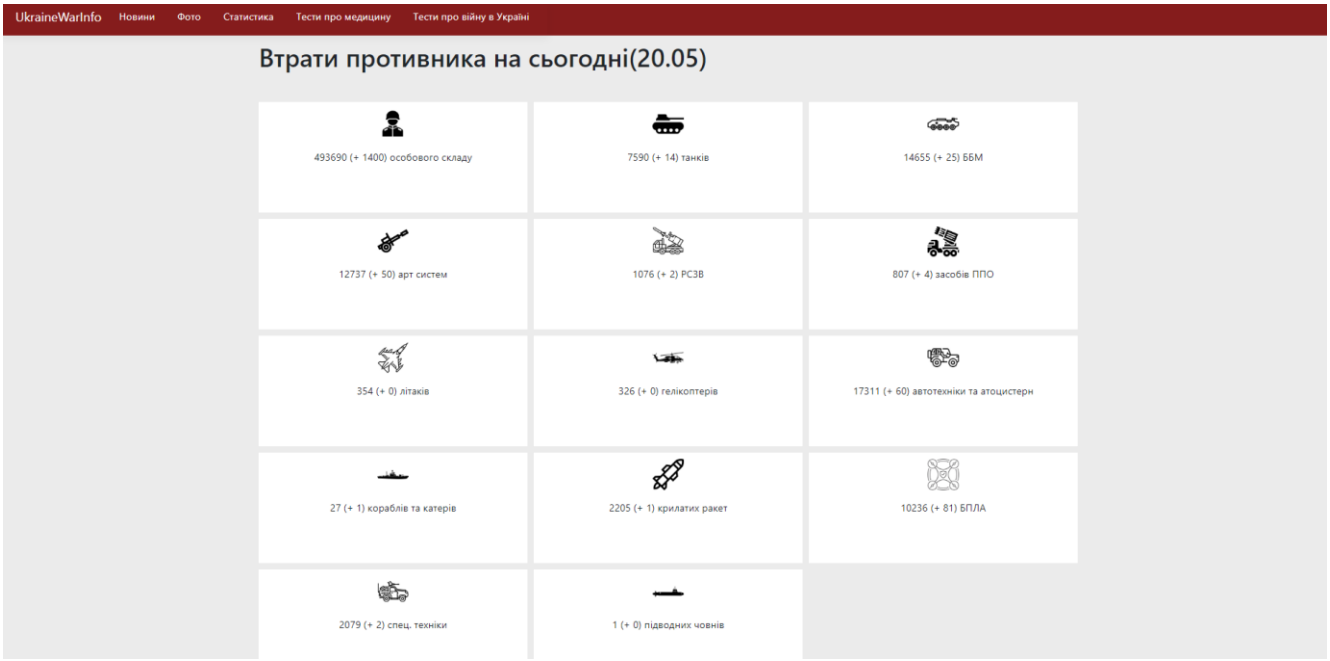


Рис. 4.5 Сторінка зі статистикою втрат ворога на фронті

4.3 Мануальне тестування

Мануальне тестування є важливим етапом перевірки функціональності та надійності веб-застосунку, що дозволяє виявити помилки, які можуть бути пропущені автоматичними тестами. Для тестування веб-ресурсу, присвяченого війні в Україні, було проведено ретельне мануальне тестування, що охоплює кілька ключових аспектів його функціонування.

Основною метою мануального тестування є перевірка коректної роботи усіх функцій веб-ресурсу. В першу чергу, тестуванню піддавалися такі функції як перегляд новин, проходження освітніх тестів, перегляд статистики та можливість сортування новин.

Перегляд новин

Однією з важливих функцій веб-ресурсу є можливість перегляду новин. Мануальне тестування підтвердило, що всі новини коректно відображаються на сторінці з переліком новин. Було перевірено, що новини відсортовані за датою публікації, починаючи з найновіших, що забезпечує користувачам доступ до актуальної інформації. Крім того, функція пагінації працює правильно у випадку перевищення обмеження на кількість новин на одній сторінці, що забезпечує зручну навігацію по ресурсу.

На рисунку 4.6 зображено інтерфейс сторінки перегляду списку новин.

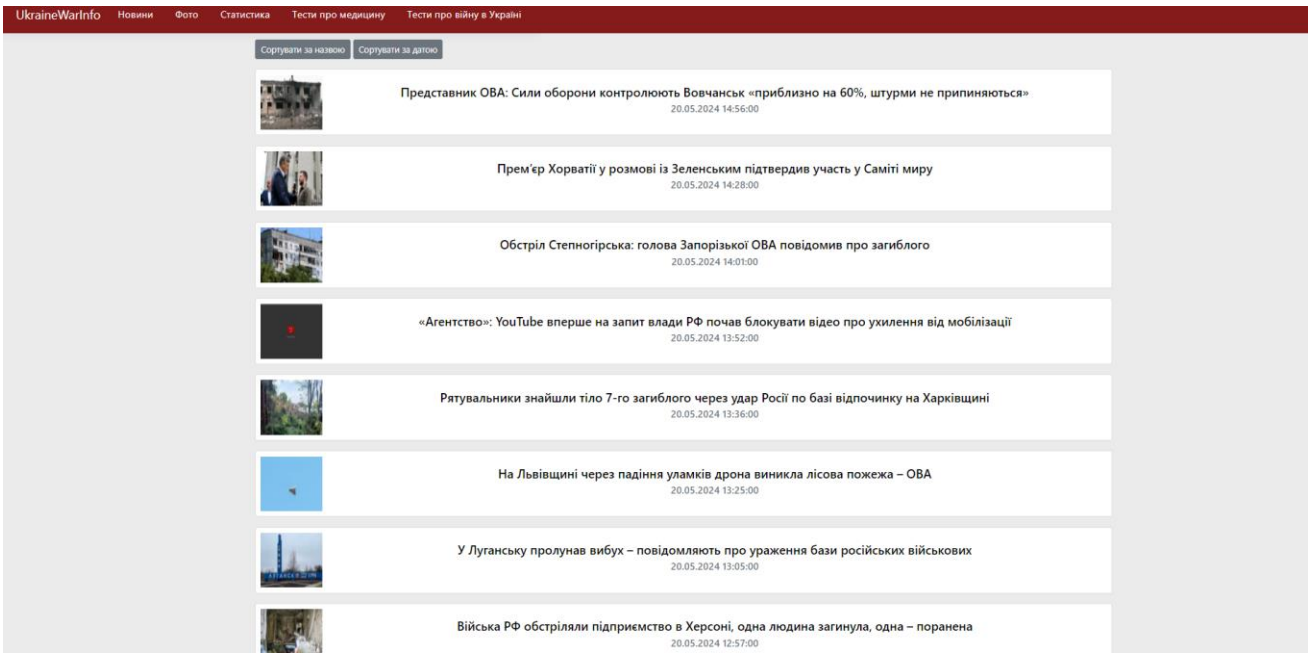


Рис. 4.6 Сторінка перегляду списку новин

Проходження освітніх тестів

Іншою ключовою функцією є можливість проходження освітніх тестів, що допомагають користувачам перевірити свої знання. Тести відображаються коректно, а результати проходження тесту відображаються правильно. Після завершення тесту користувачам надаються правильні відповіді, що дозволяє їм аналізувати свої помилки та покращувати знання.

На рисунку 4.7 зображено інтерфейс сторінки проходження тестів з тактичної медицини.

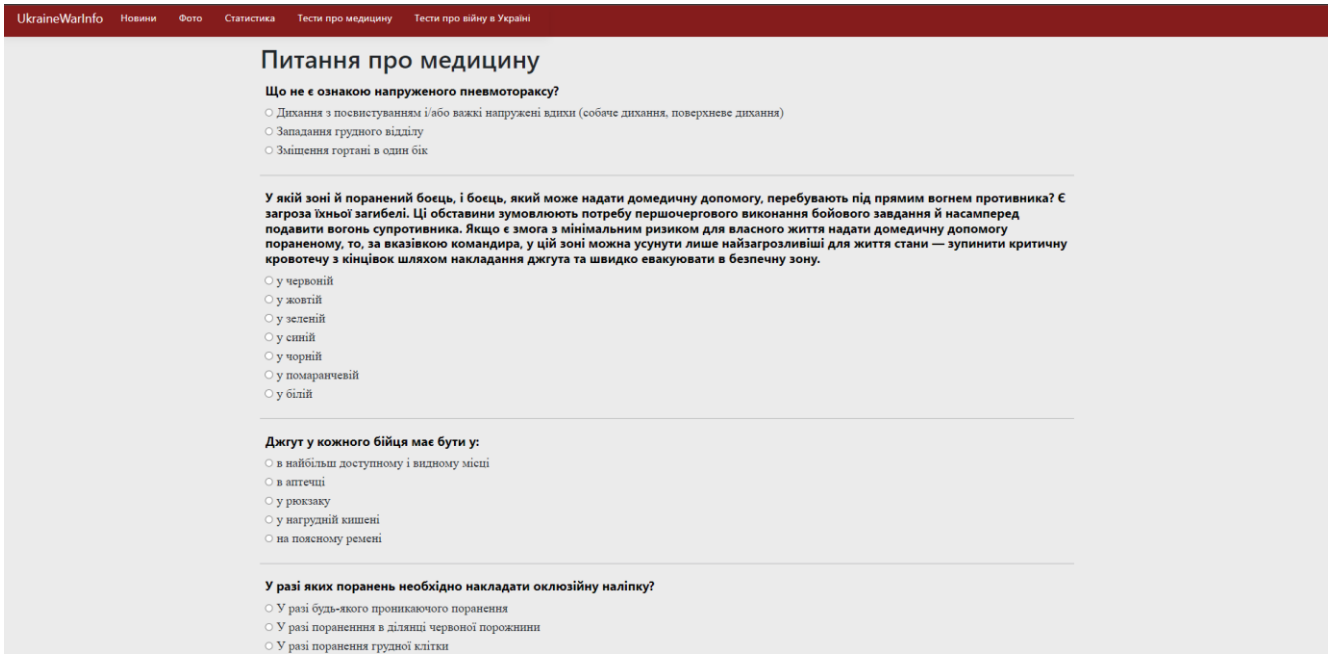


Рис. 4.7 Сторінка проходження тестів з тактичної медицини.

На рисунку 4.8 зображено інтерфейс сторінки проходження тестів на тему “Події під час війни в Україні”.

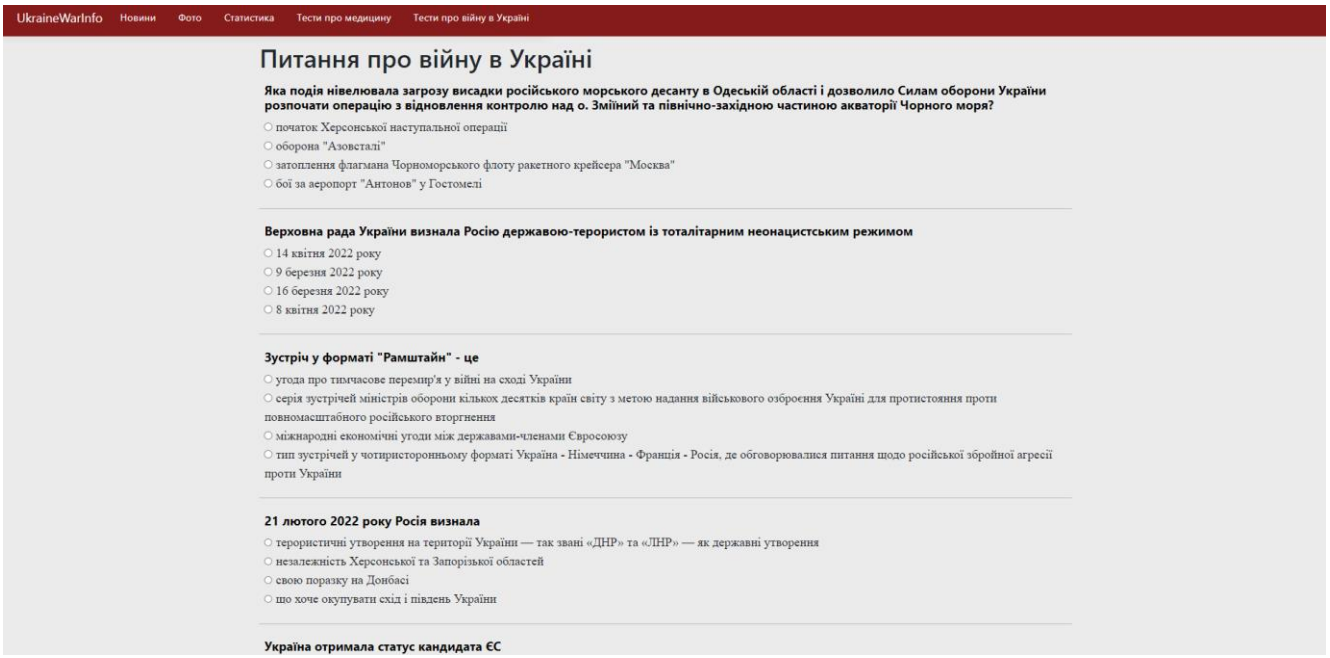


Рис. 4.8 Сторінка проходження тестів на тему “Події під час війни в Україні”.

На рисунку 4.9 зображено інтерфейс сторінки результатів проходження освітнього тесту, де зеленим кольором відображені правильні відповіді на питання.

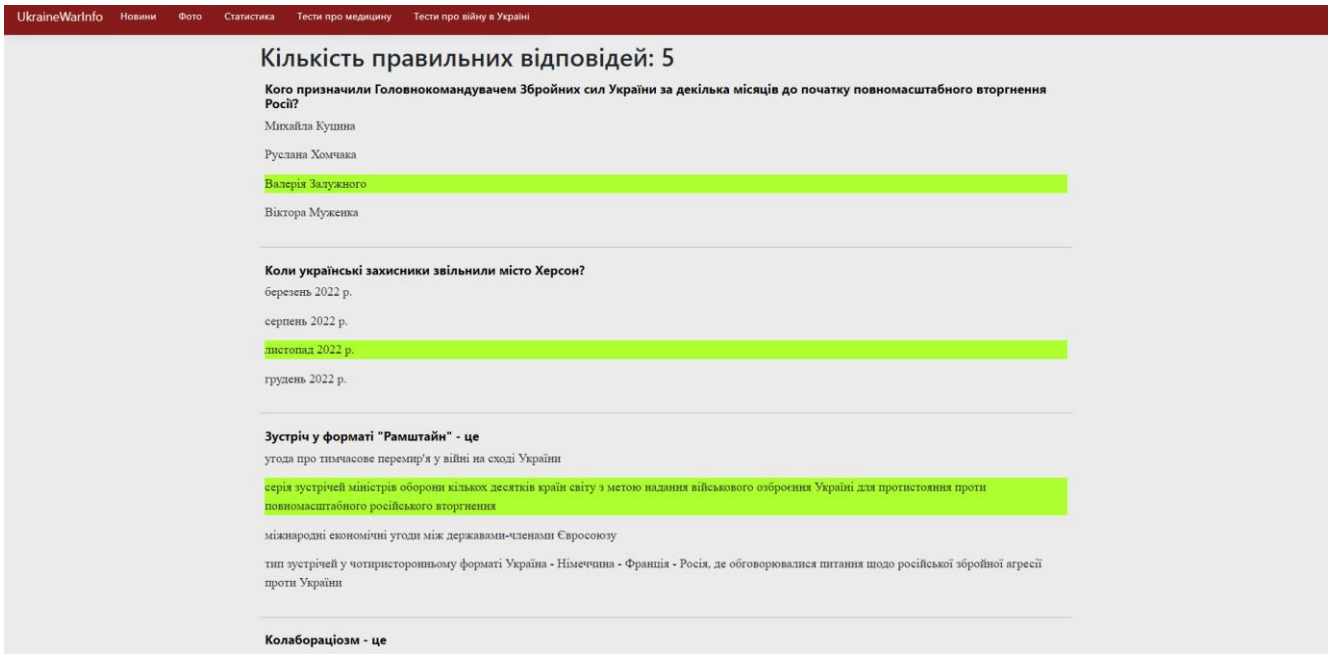


Рис. 4.9 Сторінка результатів проходження освітнього тесту

Перегляд статистики

Розділ зі статистикою втрат ворожої піхоти та техніки є важливою частиною веб-ресурсу. Мануальне тестування підтвердило, що цей розділ містить актуальну інформацію, яка коректно відображається. Це забезпечує користувачам доступ до важливих даних про стан військових дій на сьогоднішній день.

На рисунку 4.10 зображено інтерфейс сторінки статистики втрат ворога на фронті.

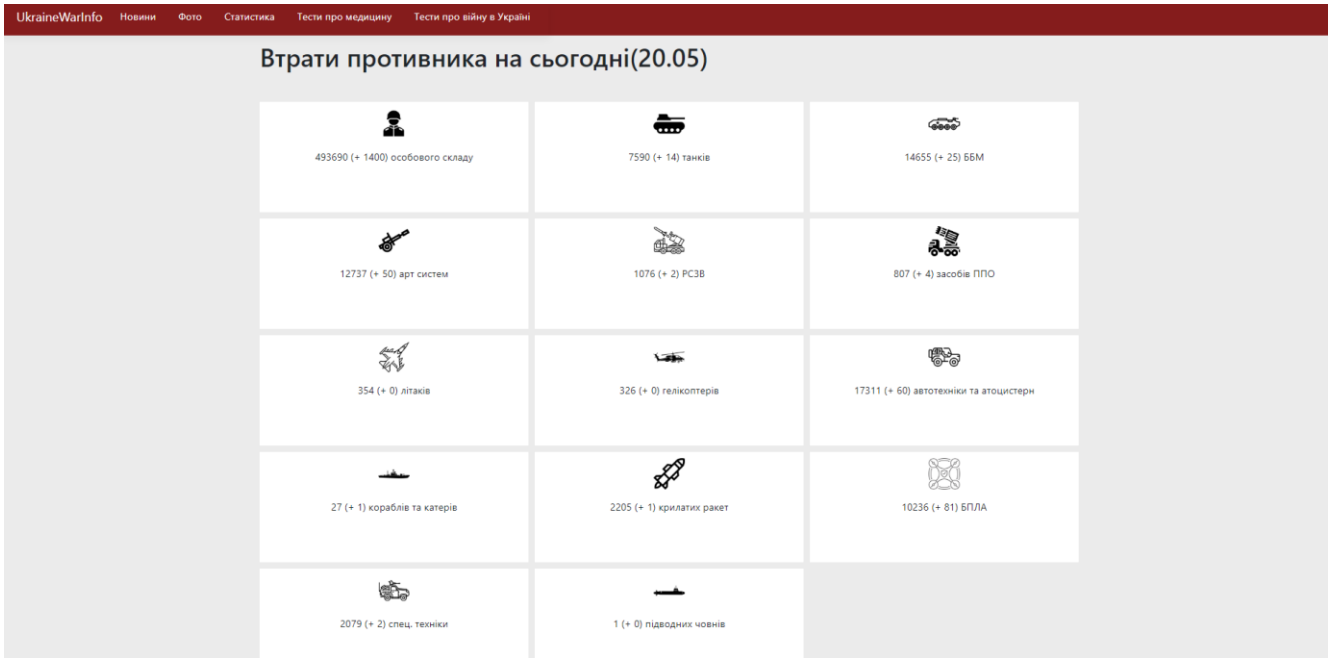


Рис. 4.10 Сторінка статистики втрат ворога на фронті

Можливість сортування новин

Сортування новин за назвою та датою є важливою функцією для зручності користувачів. Мануальне тестування показало, що всі новини коректно сортуються за вказаними параметрами, що полегшує пошук потрібної інформації.

На рисунку 4.11 зображено інтерфейс сторінки списку новин відсортованих за назвою.

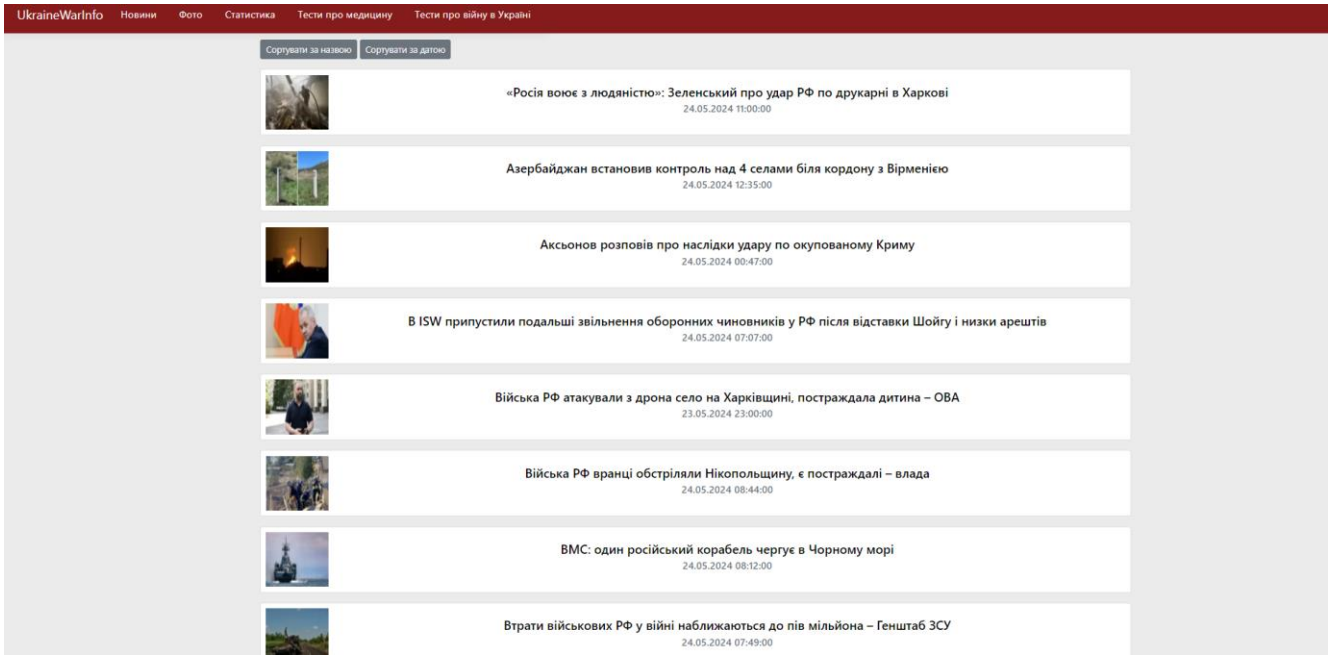


Рисунок 4.11 Сторінка списку новин відсортованих за назвою

На рисунку 4.12 зображено інтерфейс сторінки списку новин відсортованих за датою публікації.

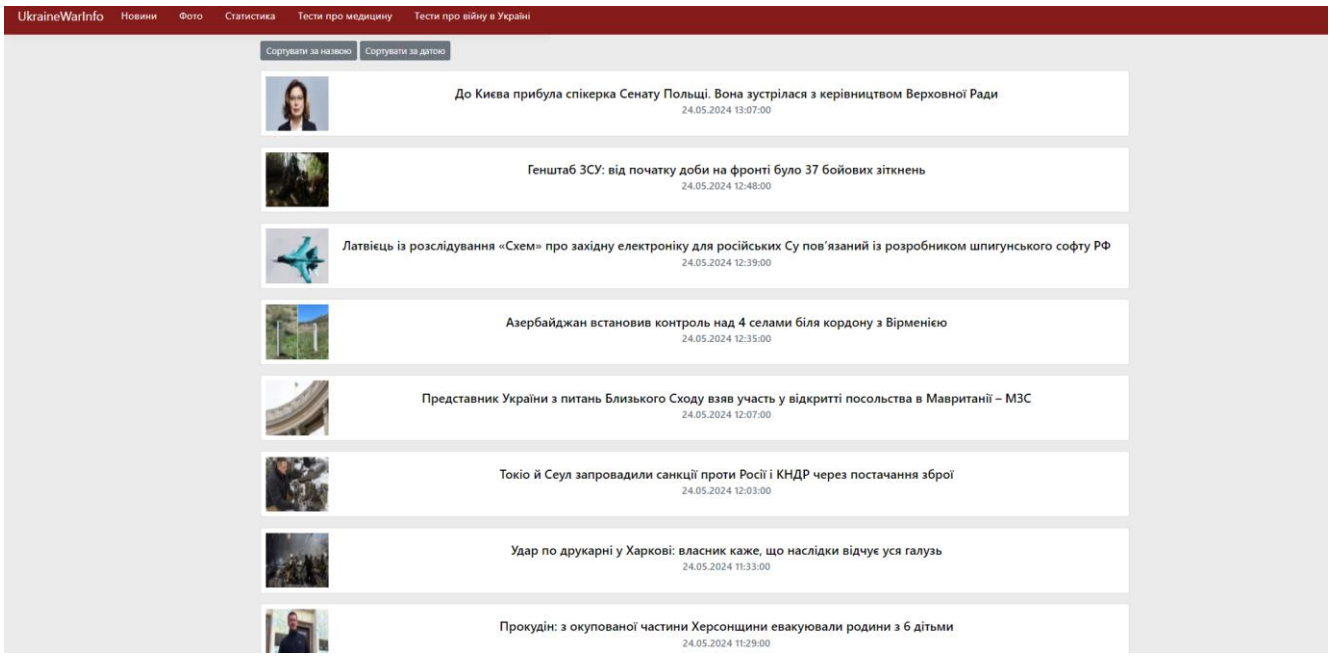


Рис. 4.12 Сторінка списку новин відсортованих за датою публікації

Загальні тести

Для забезпечення коректної роботи веб-ресурсу в різних умовах було проведено загальні тести. Перевірено, що веб-ресурс коректно відображається в різних браузерах, а всі посилання на сайті працюють правильно і ведуть на відповідні сторінки. Це гарантує, що незалежно від обраного браузера, користувачі матимуть доступ до всіх функцій веб-ресурсу.

Мануальне тестування дозволило підтвердити коректну роботу веб-ресурсу та його відповідність заявленому функціоналу. Це включає правильне відображення новин, коректну роботу освітніх тестів, актуальність статистичних даних та зручність користування функціями сортування новин. Проведене тестування забезпечує впевненість у надійності та функціональності веб-ресурсу, що є критично важливим для користувачів, які шукають достовірну інформацію про війну в Україні.

ВИСНОВКИ

1. Проведено дослідження, в ході якого було виявлено, що існуючі освітні веб-ресурси про війну в Україні мають високу актуальність для суспільства, особливо для молоді, яка не завжди обізнана про історичні, політичні та соціальні аспекти конфлікту.

2. Розроблено структуровану систему зберігання та представлення інформації, що дозволяє орієнтуватися в потоці даних та швидко знаходити необхідну інформацію.

3. Сформульовано вимоги до програмного забезпечення, що враховують попередні недоліки та пропонують шляхи їх вирішення.

4. Розроблено парсер, який автоматизує процес отримання та оновлення інформації про події в Україні.

5. Створено освітні тести, які дозволяють перевірити знання користувачів про війну в Україні та сприяють підвищенню рівня їхньої освіченості з цієї теми.

4. Проведено огляд та аналіз ІТ-засобів дозволив обрати оптимальний набір інструментів для розробки програмного забезпечення, яке задовольняє вимоги та потреби користувачів.

6. Розроблено веб-застосунок, який забезпечує користувачів доступом до освітньої інформації про війну в Україні та перевіркою знань з цієї теми.

7. Проведено мануальне тестування веб-застосунку, що дозволило виявити та виправити помилки, а також перевірити його функціональність.

8. Робота пройшла апробацію:

Конєв А.О., Замрій І.В. Визначення вимог до освітнього веб-сервісу про війну в Україні. *Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях».* 24 квітня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С. 164-165.

Конєв А.О., Замрій І.В. Визначення додаткових функцій освітнього веб-сервісу про війну в Україні. *IV Всеукраїнська Науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті».* 15 травня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К: ДУІКТ, 2024. С. 64-66.

ПЕРЕЛІК ПОСИЛАНЬ

1. ASP.NET [Електронний ресурс] // Microsoft – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-8.0.
2. HtmlAgilityPack [Електронний ресурс] // Jonathan Magnan – Режим доступу до ресурсу: https://html-agility-pack.net/documentation.
3. .NET [Електронний ресурс] // Microsoft – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/dotnet/.
4. Entity framework [Електронний ресурс] // Microsoft – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/ef/.
5. Nachasi [Електронний ресурс] – Режим доступу до ресурсу: https://nachasi.com/.
6. Тиждень [Електронний ресурс] – Режим доступу до ресурсу: http://www.week.dp.gov.ua/.
7. Накипіло [Електронний ресурс] – Режим доступу до ресурсу: https://nakipelo.ua/.
8. The Model View Controller Pattern – MVC Architecture and Frameworks Explained [Електронний ресурс] // Rafael D. Hernandez. – 2021. – Режим доступу до ресурсу: https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/.
9. SQLite Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.sqlitetutorial.net/.
10. You’re All Doing Entity Framework Wrong [Електронний ресурс] // Michael Hoagland. – 2017. – Режим доступу до ресурсу: https://medium.com/@hoagsie/youre-all-doing-entity-framework-wrong-ea0c40e20502.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ
(Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка освітнього Web-ресурсу про війну в Україні мовою C#

Виконав студент 4 курсу
Групи ПД-41
Конєв Антон Олександрович
Керівник роботи
Д.т.н, доцент, завідувач кафедри ІПЗ Замрій Ірина Вікторівна

Київ – 2024

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** створення освітніх матеріалів для шкіл, університетів і самостійного навчання, що допоможуть глибше зрозуміти історичні та соціальні аспекти війни, сприяючи формуванню критичного мислення та громадянської свідомості, а також забезпечення доступу до структурованої інформації про події, пов'язані з війною в Україні, для широкого кола користувачів.
- **Об’єкт дослідження** є процес підвищення обізнаності та самонавчання подіям в Україні.
- **Предмет дослідження** є освітній веб-ресурс для підвищення рівня обізнаності населення про події в Україні.

2

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Проаналізувати аналоги інформаційних ресурсів про війну в Україні.
2. Підбір офіційних джерел для збору даних.
3. Розробити інтерфейс користувача для навігації та отримання інформації.
4. Проаналізувати застосунки з освітніми тестами про війну в Україні.
5. Розробити власний парсер для отримання інформації про події в Україні з інформаційних ресурсів.
6. Розробити освітні тести.
7. Спроектувати та розробити веб-застосунок для доступу до освітньої інформації про війну в Україні.
8. Провести тестування застосунку.

3

АНАЛІЗ АНАЛОГІВ

	Парсер новин	Доступність через різні пристрої	Наявність сортування новин	Проходження освітніх тестів
nachasi.com	-	+	-	+
week.dp.gov.ua	-	+	-	+
nakipelo.ua	-	+	-	+
Ukrainewarinfo	+	+	+	+

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги:

- *Перегляд новин. Користувачі можуть переглядати список новин,відомі на сьогоднішній день;*
- *Проходження освітніх тестів. Користувачі можуть перевірити свою обізнаність в сфері “Війна в Україні”.*
- *Перегляд статистики. Користувачі можуть переглядати актуальну статистику втрат ворожої піхоти та техніки;*
- *Можливість сортування новин. Користувачі можуть сортувати новини за заголовком та датою;*

Нефункціональні вимоги:

- *Сумісність: сумісність з різними пристроями та браузерами для забезпечення доступності для широкого кола користувачів;*
- *Відповідність контенту: Контент повинен відповідати етичним стандартам та не порушувати законодавства України.*

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

HtmlAgilityPack

ASP.NET

C#

HTML

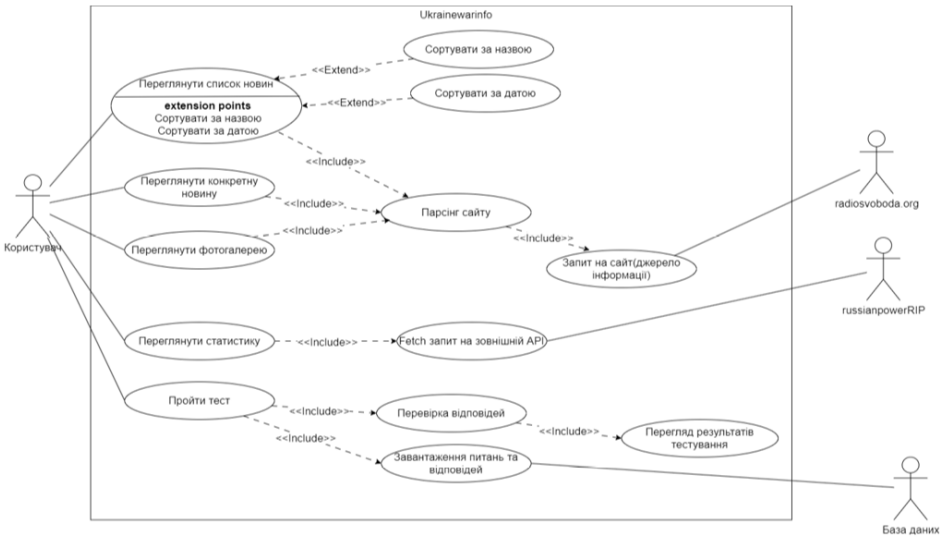
CSS

Bootstrap

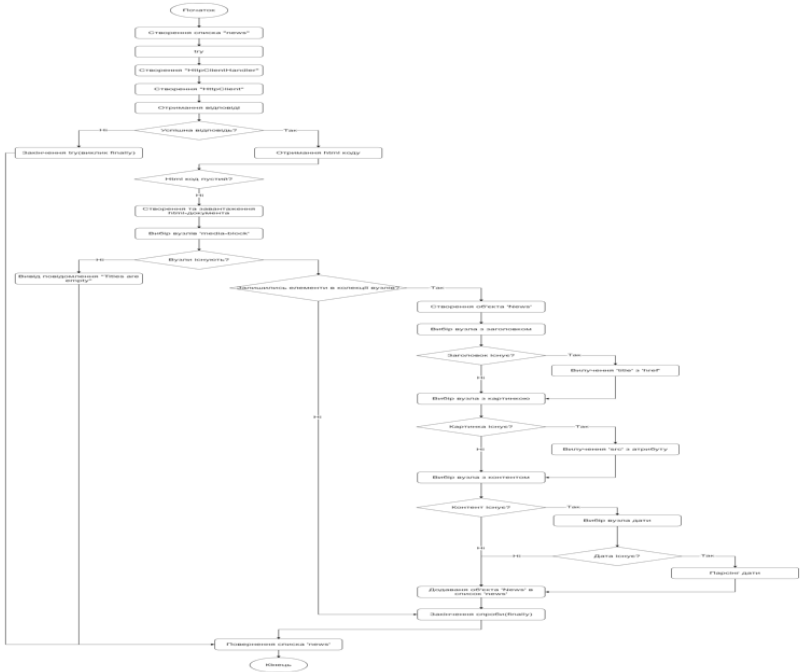
JS

6

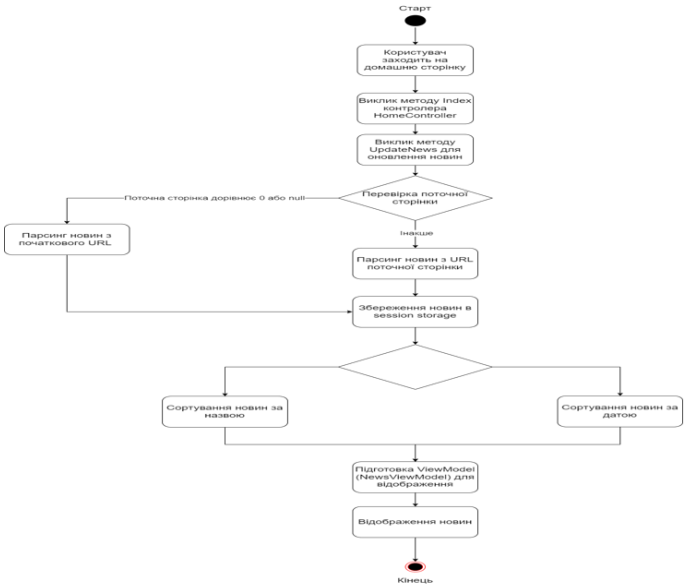
Діаграма варіантів використання



Блок-схема роботи парсеру

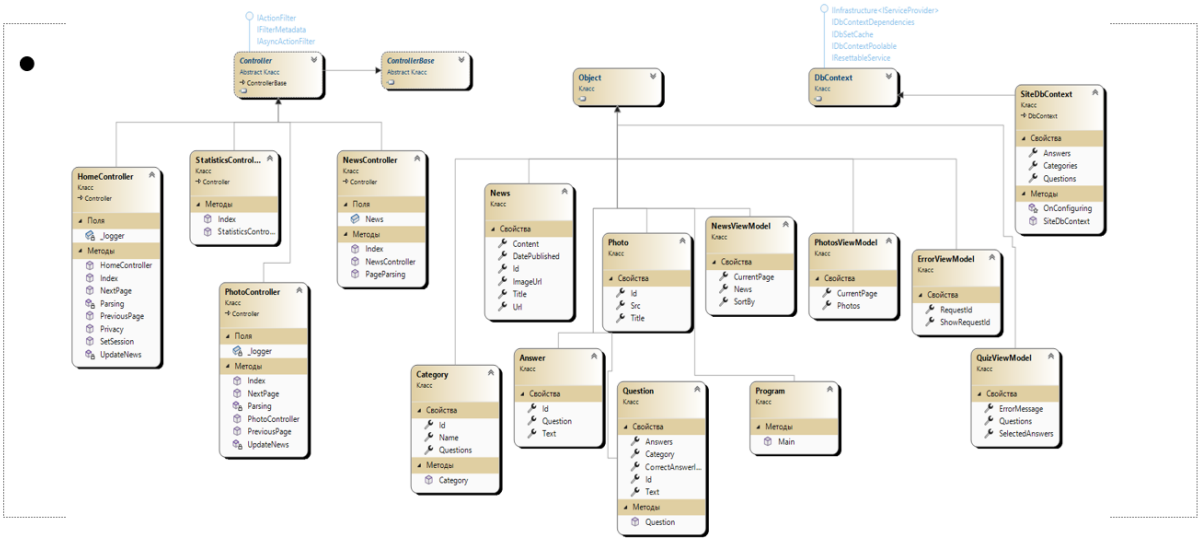


Діаграма діяльності(перегляд новин)



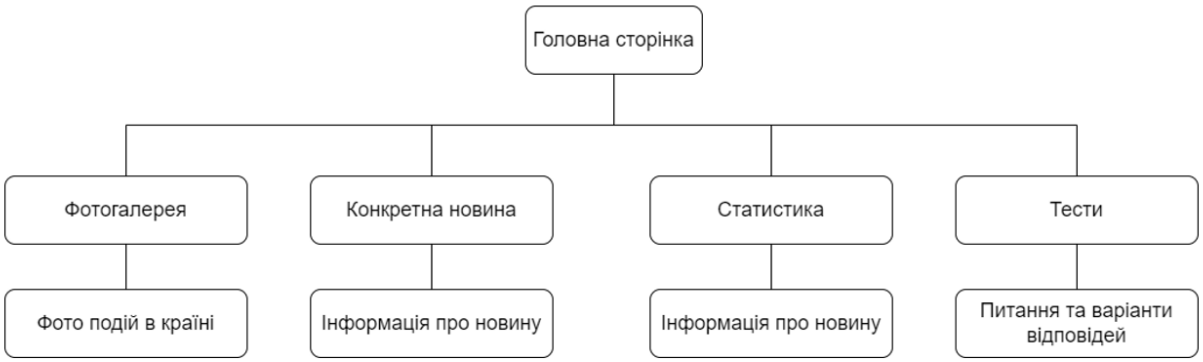
9

Діаграма класів



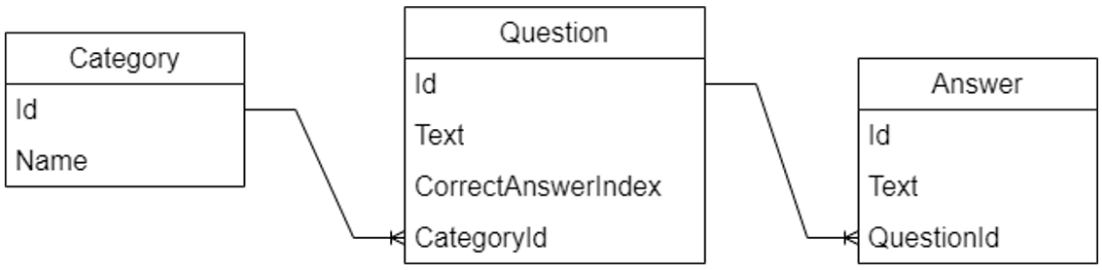
10

Мапа сайту



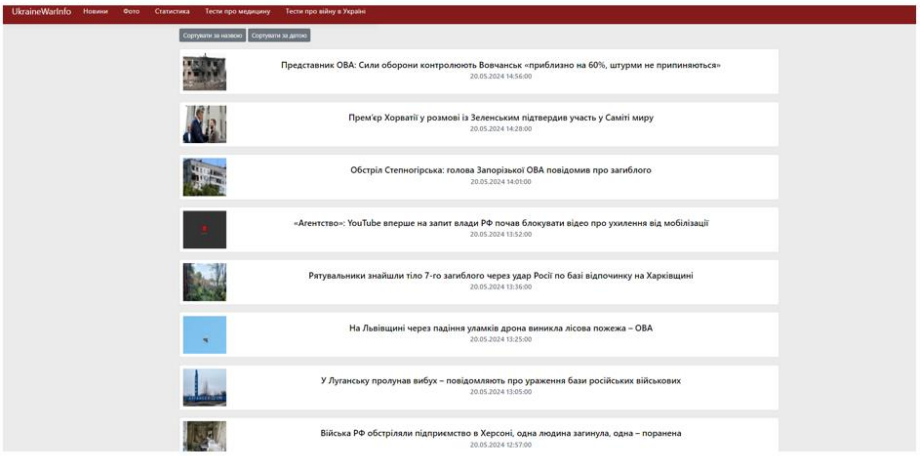
11

Діаграма бази даних



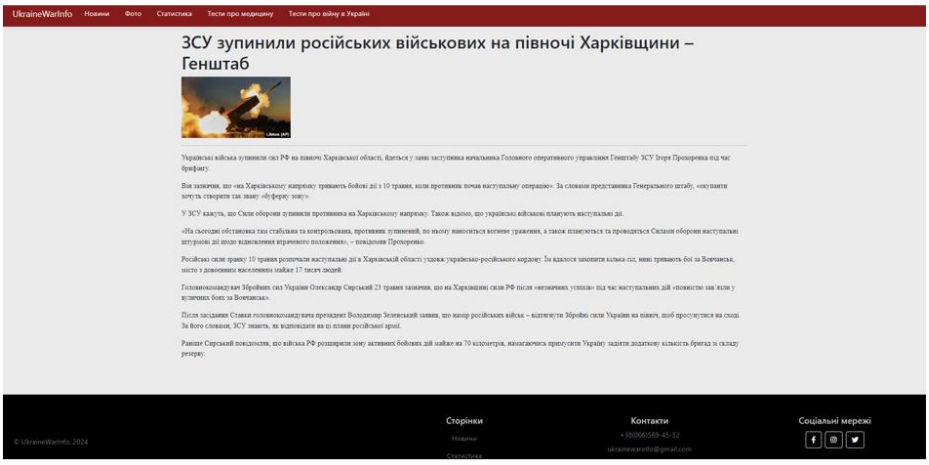
12

ЕКРАННІ ФОРМИ



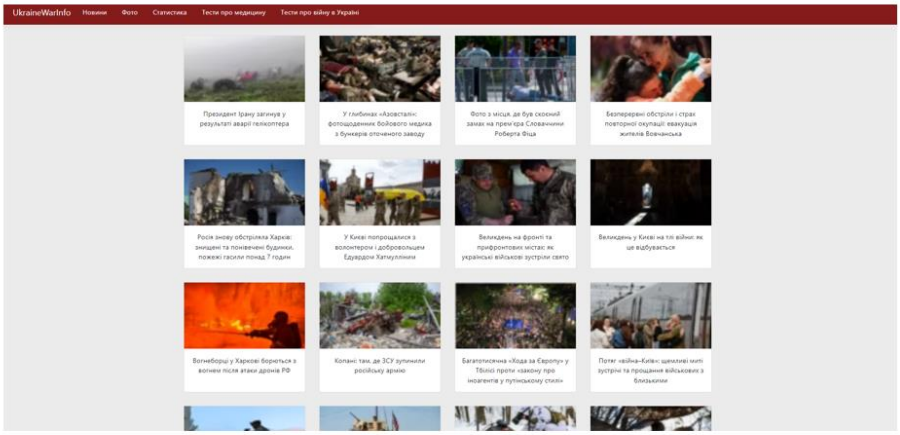
Головна сторінка

ЕКРАННІ ФОРМИ



Сторінка з конкретною новиною

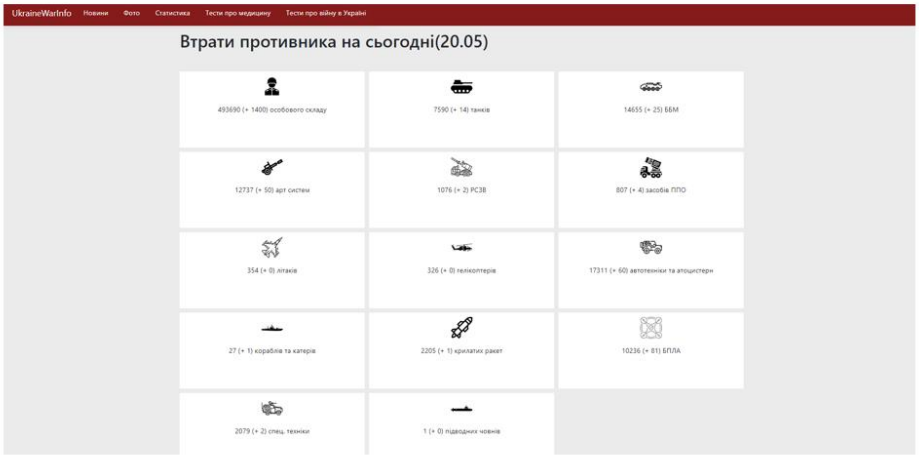
ЕКРАННІ ФОРМИ



Сторінка фотогалереї

15

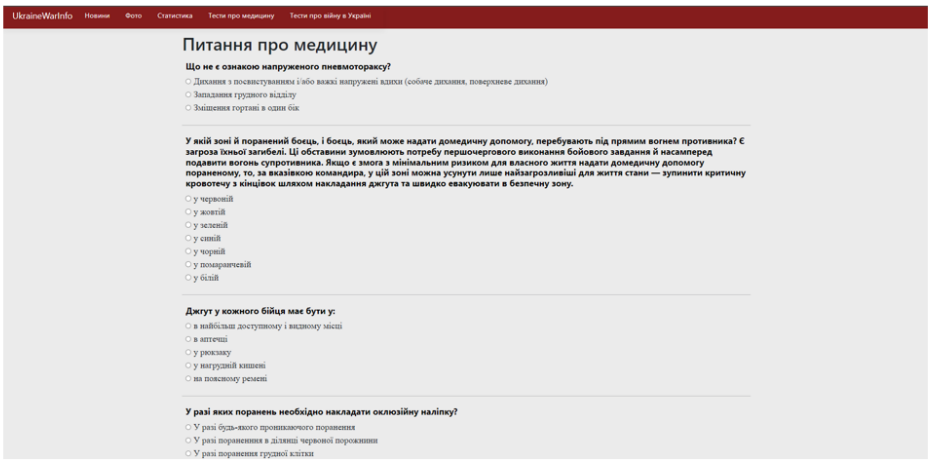
ЕКРАННІ ФОРМИ



Сторінка статистики втрат ворога на фронті

16

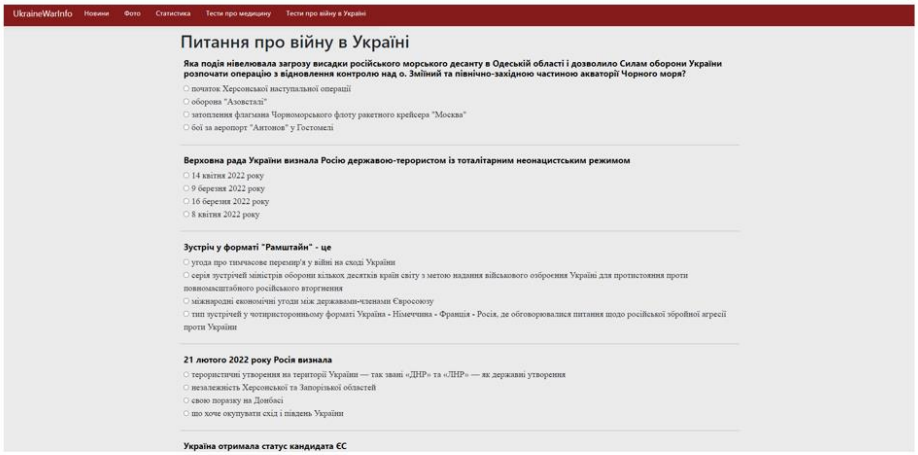
ЕКРАННІ ФОРМИ



Сторінка тестів про медицину

17

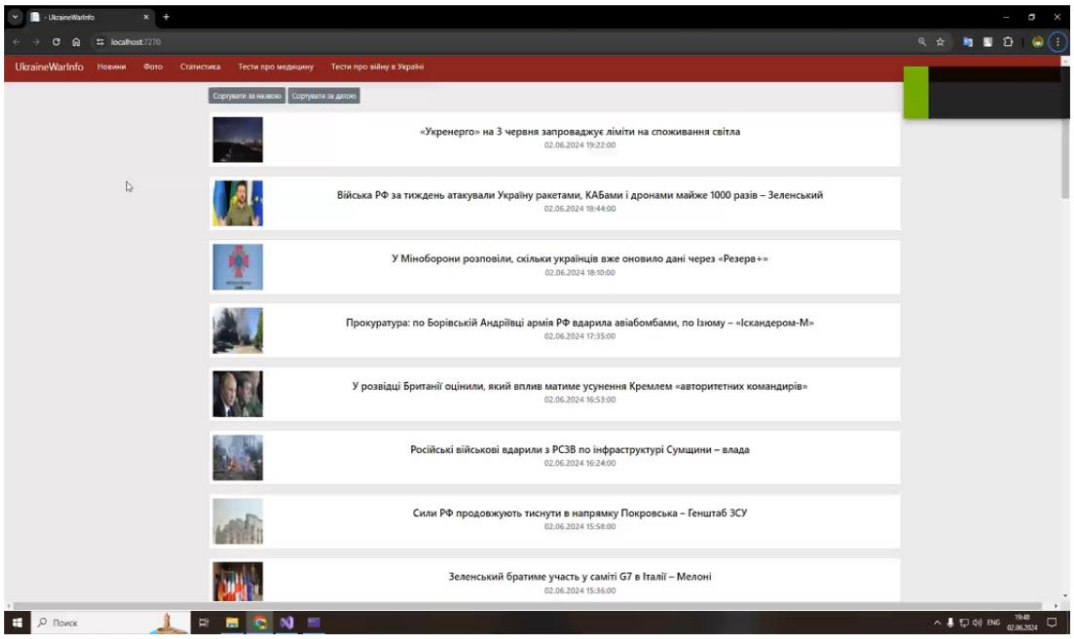
ЕКРАННІ ФОРМИ



Сторінка тестів про війну в Україні

18

ЕКРАННІ ФОРМИ



АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Конєв А.О., Замрій І.В. Визначення вимог до освітнього веб-сервісу про війну в Україні. Всеукраїнська науково-технічна конференція “Застосування програмного забезпечення в інформаційно-комунікаційних технологіях”. 24 квітня 2024р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С. 164 - 165.
2. Конєв А.О., Замрій І.В. Визначення додаткових функцій освітнього веб-сервісу про війну в Україні. IV Всеукраїнська Науково-практична конференція “Сучасні інтелектуальні інформаційні технології в науці та освіті”. 15 травня 2024р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К: ДУІКТ, 2024. С. 64 - 66.

ВИСНОВКИ

1. Проведено дослідження, в ході якого було виявлено, що існуючі освітні веб-ресурси про війну в Україні мають високу актуальність для суспільства, особливо для молоді, яка не завжди обізнана про історичні, політичні та соціальні аспекти конфлікту.
2. Розроблено структуровану систему зберігання та представлення інформації, що дозволяє орієнтуватися в потоці даних та швидко знаходити необхідну інформацію.
3. Сформульовано вимоги до програмного забезпечення, що враховують попередні недоліки та пропонують шляхи їх вирішення.
4. Розроблено парсер, який автоматизує процес отримання та оновлення інформації про події в Україні.
5. Створено освітні тести, які дозволяють перевірити знання користувачів про війну в Україні та сприяють підвищенню рівня їхньої освіченості з цієї теми.
4. Проведено огляд та аналіз IT-засобів дозволив обрати оптимальний набір інструментів для розробки програмного забезпечення, яке задовольняє вимоги та потреби користувачів.
6. Розроблено веб-застосунок, який забезпечує користувачів доступом до освітньої інформації про війну в Україні та перевіркою знань з цієї теми.
7. Проведено мануальне тестування веб-застосунку, що дозволило виявити та виправити помилки, а також перевірити його функціональність.

ДОДАТОК Б. ЛІСТИНГИ ОСНОВНИХ МОДУЛІВ

HomeController.

```
Ссылок: 5
public class HomeController : Controller
{
    private readonly ILogger<HomeController> _logger;

    Ссылок: 0
    public HomeController(ILogger<HomeController> logger)
    {

    }

    Ссылок: 3
    private void UpdateNews()
    {
        int? currentPage = HttpContext.Session.GetInt32("CurrentPage");
        List<News> news = new List<News>();

        if (currentPage == null || currentPage == 0)
        {
            news = Parsing(url: "https://www.radiosvoboda.org/z/630");
        }
        else
        {
            news = Parsing(url: $"https://www.radiosvoboda.org/z/630?p={currentPage}");
        }
        News lastNews = news.Last();

        HttpContext.Session.SetString("News", JsonSerializer.Serialize(news));
        HttpContext.Session.SetString("LastNews", JsonSerializer.Serialize(lastNews));
    }

    Ссылок: 0
    public IActionResult Index(string sortBy)
    {
```

```
Ссылок: 0
public IActionResult Index(string sortBy)
{
    UpdateNews();
    string newsJson = HttpContext.Session.GetString("News");
    List<News> news;

    if (!string.IsNullOrEmpty(newsJson))
    {
        news = JsonSerializer.Deserialize<List<News>>(newsJson);
    }
    else
    {
        news = new List<News>();
    }

    if (!string.IsNullOrEmpty(sortBy))
    {
        if (sortBy.Equals("Date"))
        {
            news = news.OrderByDescending(n => n.DatePublished).ToList();

        }
        else if (sortBy.Equals("Title"))
        {

            news = news.OrderBy(n => n.Title).ToList();

        }
    }

    int? currentPage = HttpContext.Session.GetInt32("CurrentPage");
```

```
        }

        int? currentPage = HttpContext.Session.GetInt32("CurrentPage");

        var newsViewModel = new NewsViewModel
        {
            News = news,
            CurrentPage = currentPage,
            SortBy = sortBy
        };

        return View(newsViewModel);
    }

    Ссылок: 2
    private List<News> Parsing(string url)
    {
        List<News> news = new List<News>();
        try
        {
            using (HttpClientHandler hdl = new HttpClientHandler { AllowAutoRedirect = false, AutomaticDecompression = System.Net.DecompressionMethod
            {
                using (var client = new HttpClient(hdl))
                {
                    using (HttpResponseMessage response = client.GetAsync(url).Result)
                    {
                        if (response.IsSuccessStatusCode)
                        {

                            var html = response.Content.ReadAsStringAsync().Result;
                            if (!string.IsNullOrEmpty(html))
                            {
                                HtmlAgilityPack.HtmlDocument doc = new HtmlAgilityPack.HtmlDocument();
```

```
                {
                    HtmlAgilityPack.HtmlDocument doc = new HtmlAgilityPack.HtmlDocument();
                    doc.LoadHtml(html);

                    var infos = doc.DocumentNode.SelectNodes(".//div[@class='media-block ']");
                    if (infos != null && infos.Count > 0)
                    {
                        foreach (var info in infos)
                        {
                            var item = new News();
                            var titleNode = info.SelectSingleNode(".//a");
                            var imageNode = info.SelectSingleNode(".//img");
                            var contentNode = info.SelectSingleNode(".//div[contains(@class, 'media-block__content')]");
                            if (titleNode != null)
                            {
                                item.Title = HttpUtility.HtmlDecode(titleNode.Attributes["title"].Value);
                                item.Url = titleNode.Attributes["href"].Value;
                            }
                            if (imageNode != null)
                            {
                                item.ImageUrl = imageNode.Attributes["src"].Value;
                            }
                            if (contentNode != null)
                            {
                                var dateNode = contentNode.SelectSingleNode(".//span");
                                if (dateNode != null)
                                {
                                    DateTime date;
                                    if (DateTime.TryParseExact(dateNode.InnerText, "HH:mm, dd MMMM yyyy", new CultureInfo("uk-UA"), DateTimeStyles.None, out date))
                                    {
                                        item.DatePublished = date;
                                    }
                                }
                            }


                            news.Add(item);
                        }
                    }
```

```
                    }
                    else
                    {
                        Console.WriteLine("Titles are empty");
                    }
                }
            }
        }

    }
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
return news;
}

[HttpPost]
Ссылок: 0
public IActionResult SetSession(int currentPage)
{
    HttpContext.Session.SetInt32("CurrentPage", currentPage);
    return Ok();
}

[HttpPost]
Ссылок: 0
public IActionResult NextPage()
{
    int currentPage = HttpContext.Session.GetInt32("CurrentPage") ?? 0;
    currentPage++;
    HttpContext.Session.SetInt32("CurrentPage", currentPage);

    UpdateNews();

    return RedirectToAction("Index");
}
[HttpPost]
```

```
public IActionResult SetSession(int currentPage)
{
    HttpContext.Session.SetInt32("CurrentPage", currentPage);
    return Ok();
}

[HttpPost]
Ссылок: 0
public IActionResult NextPage()
{
    int currentPage = HttpContext.Session.GetInt32("CurrentPage") ?? 0;
    currentPage++;
    HttpContext.Session.SetInt32("CurrentPage", currentPage);

    UpdateNews();

    return RedirectToAction("Index");
}
[HttpPost]
Ссылок: 0
public IActionResult PreviousPage()
{
    int currentPage = HttpContext.Session.GetInt32("CurrentPage") ?? 0;
    currentPage--;
    HttpContext.Session.SetInt32("CurrentPage", currentPage);

    UpdateNews();

    return RedirectToAction("Index");
}

Ссылок: 0
public IActionResult Privacy()
{
    return View();
}
}
}
```

NewsController.

```
namespace Diplom.Controllers
{
    Ссылок: 1
    public class NewsController : Controller
    {
        public News News;
        Ссылок: 0
        public NewsController()
        {
            News = new News();

        }
        [HttpGet]
        Ссылок: 0
        public IActionResult Index(string url)
        {
            PageParsing(url);
            return View(News);
        }

        Ссылок: 1
        public void PageParsing(string url)
        {
            try
            {
                using (HttpClientHandler hdl = new HttpClientHandler { AllowAutoRedirect = false, AutomaticDecompression = System.Net.DecompressionMethods.Deflate | System.Net.Decompression
                {
                    using (var client = new HttpClient(hdl))
                    {
                        using (HttpResponseMessage response = client.GetAsync(url).Result)
                        {
                            if (response.IsSuccessStatusCode)
                            {

                                var html = response.Content.ReadAsStringAsync().Result;
                                if (!string.IsNullOrEmpty(html))
                                {
                                    HtmlAgilityPack.HtmlDocument doc = new HtmlAgilityPack.HtmlDocument();
                                    doc.LoadHtml(html);
```

```
                                    var headerNode = doc.DocumentNode.SelectSingleNode(".//div[@class='hdr-container']");
                                    if(headerNode != null)
                                    {
                                        var titleNode = headerNode.SelectSingleNode(".//h1[contains(@class, 'title')]");
                                        if (titleNode != null)
                                        {
                                            News.Title = HttpUtility.HtmlDecode(titleNode.InnerText);
                                        }
                                        var dateNode = headerNode.SelectSingleNode(".//time");
                                        if (dateNode != null)
                                        {
                                            DateTime date;
                                            if (DateTime.TryParseExact(dateNode.InnerText, "HH:mm, dd MMMM yyyy", new CultureInfo("uk-UA"), DateTimeStyles.None, out date))
                                            {
                                                News.DatePublished = date;
                                            }
                                        }

                                        var imageNode = headerNode.SelectSingleNode(".//img");
                                        if(imageNode != null)
                                        {
                                            News.ImageUrl = imageNode.Attributes["src"].Value;
                                        }

                                    }
                                    var bodyNode = doc.DocumentNode.SelectSingleNode(".//div[contains(@class, 'body-container')]");
                                    if(bodyNode != null)
                                    {
                                        var contentNode = bodyNode.SelectSingleNode(".//div[contains(@class, 'wsw')]");
                                        if (contentNode != null)
                                        {
                                            News.Content = new List<string>();
                                            var contents = contentNode.SelectNodes(".//p");
                                            foreach (var content in contents)
                                            {
                                                News.Content.Add(content.InnerText);
                                            }
                                        }
                                    }

                                }
                            }
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
```

PhotoController.

```
public class PhotoController : Controller
{
    private readonly ILogger<HomeController> _logger;

    Ссылок: 0
    public PhotoController(ILogger<HomeController> logger)
    {

    }

    Ссылок: 3
    private void UpdateNews()
    {
        int? currentPage = HttpContext.Session.GetInt32("CurrentPhotoPage");
        List<Photo> photos = new List<Photo>();

        if (currentPage == null || currentPage == 0)
        {
            photos = Parsing(url: "https://www.radiosvoboda.org/z/2963");
        }
        else
        {
            photos = Parsing(url: $"https://www.radiosvoboda.org/z/2963?p={currentPage}");
        }
        Photo lastPhoto = photos.Last();

        HttpContext.Session.SetString("Photos", JsonSerializer.Serialize(photos));
        HttpContext.Session.SetString("LastPhoto", JsonSerializer.Serialize(lastPhoto));
    }

    Ссылок: 0
    public IActionResult Index()
    {
        UpdateNews();
        string pjotosJson = HttpContext.Session.GetString("Photos");
        List<Photo> photos;
```

```
        if (!string.IsNullOrEmpty(pjotosJson))
        {
            photos = JsonSerializer.Deserialize<List<Photo>>(pjotosJson);
        }
        else
        {
            photos = new List<Photo>();
        }

        int currentPage = HttpContext.Session.GetInt32("CurrentPhotoPage") ?? 0;

        var photosViewModel = new PhotosViewModel
        {
            Photos = photos,
            CurrentPage = currentPage,
        };

        return View(photosViewModel);
    }

    Ссылок: 2
    private List<Photo> Parsing(string url)
    {
        List<Photo> photos = new List<Photo>();
        try
        {
            using (HttpClientHandler hdl = new HttpClientHandler { AllowAutoRedirect = false, AutomaticDecompression = System.Net.DecompressionMethods.Deflate | System.Net.Decompression
            {
                using (var client = new HttpClient(hdl))
                {
                    using (HttpResponseMessage response = client.GetAsync(url).Result)
                    {
                        if (response.IsSuccessStatusCode)
                        {

                            var html = response.Content.ReadAsStringAsync().Result;
                            if (!string.IsNullOrEmpty(html))
                            {
                                HtmlAgilityPack.HtmlDocument doc = new HtmlAgilityPack.HtmlDocument();
                                doc.LoadHtml(html);
```

```
                                doc.LoadHtml(html);

                                var infos = doc.DocumentNode.SelectNodes(".//div[@class='media-block ']");
                                if (infos != null && infos.Count > 0)
                                {
                                    foreach (var info in infos)
                                    {
                                        var item = new Photo();
                                        var titleNode = info.SelectSingleNode(".//a");
                                        var imageNode = info.SelectSingleNode(".//img");
                                        if (titleNode != null)
                                        {
                                            item.Title = HttpUtility.HtmlDecode(titleNode.Attributes["title"].Value);
                                        }
                                        if (imageNode != null)
                                        {
                                            item.Src = imageNode.Attributes["src"].Value;
                                        }


                                        photos.Add(item);
                                    }

                                }
                                else
                                {
                                    Console.WriteLine("Titles are empty");
                                }
                            }
                        }
                    }

                }
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
        return photos;
    }

    [HttpGet]
    Ссылок: 0
```

```
                            }
                        }
                    }
                }

            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
        return photos;
    }

    [HttpGet]
    Ссылок: 0
    public IActionResult NextPage()
    {
        int currentPage = HttpContext.Session.GetInt32("CurrentPhotoPage") ?? 0;
        currentPage++;
        HttpContext.Session.SetInt32("CurrentPhotoPage", currentPage);

        UpdateNews();

        return RedirectToAction("Index");
    }
    [HttpGet]

    public IActionResult PreviousPage()
    {
        int currentPage = HttpContext.Session.GetInt32("CurrentPhotoPage") ?? 0;
        currentPage--;
        HttpContext.Session.SetInt32("CurrentPhotoPage", currentPage);

        UpdateNews();

        return RedirectToAction("Index");
    }
}
}
```

StatisticsController.

```
namespace Diplom.Controllers
{
    Ссылок: 1
    public class StatisticsController : Controller
    {
        Ссылок: 0
        public StatisticsController() { }

        Ссылок: 0
        public ActionResult Index()
        {
            return View();
        }


    }
}
```

TestController.

```
Ссылок: 1
public class TestController : Controller
{
    private readonly SiteDbContext _siteDbContext;

    Ссылок: 0
    public TestController(SiteDbContext siteDbContext)
    {
        _siteDbContext = siteDbContext;
    }

    Ссылок: 0
    public ActionResult Index()
    {
        return View();
    }

    Ссылок: 2
    public List<Question> ShuffleQuestions(string category)
    {
        var rand = new Random();
        var categoryId = _siteDbContext.Categories.Where(c => c.Name == category).Select(c => c.Id).FirstOrDefault();
        var questions = _siteDbContext.Questions
            .Include(q => q.Answers)
            .Where(q => q.Category.Id == categoryId)
            .ToList();

        int num = rand.Next(1, questions.Count + 1);
        questions = questions.OrderBy(q => rand.Next(1, questions.Count + 1)).Take(5).ToList();

        return questions;
    }

    Ссылок: 0
    public ActionResult Medicine(string category)
    {
```

```
    Ссылок: 0
    public ActionResult Medicine(string category)
    {
        var questions = ShuffleQuestions(category);
        var model = new QuizViewModel
        {
            Questions = questions,
            SelectedAnswers = new List<int>(),
            ErrorMessage = string.Empty
        };

        ViewData["QuestionTitle"] = "медицину";

        return View("Questions", model);
    }

    Ссылок: 0
    public ActionResult History(string category)
    {
        var questions = ShuffleQuestions(category);
        var model = new QuizViewModel
        {
            Questions = questions,
            SelectedAnswers = new List<int>(),
            ErrorMessage = string.Empty
        };

        ViewData["QuestionTitle"] = "війну в Україні";

        return View("Questions", model);
    }

    [HttpPost]
    Ссылок: 0
    public ActionResult CheckAnswers(List<int> QuestionId, List<int> selectedAnswers)
    {
```

```
public ActionResult CheckAnswers(List<int> QuestionId, List<int> selectedAnswers)
{
    var questions = _siteDbContext.Questions.Include(q => q.Answers).ToList();

    if (QuestionId == null || selectedAnswers == null || QuestionId.Count != selectedAnswers.Count)
    {
        var modelWithErrors = new QuizViewModel
        {
            Questions = _siteDbContext.Questions.Include(q => q.Answers).Where(q => QuestionId.Contains(q.Id)).ToList(),
            SelectedAnswers = selectedAnswers ?? new List<int>(),
            ErrorMessage = "Виберіть відповіді на всі питання."
        };
        return View("Questions",modelWithErrors);
    }

    int correctAnswersCount = 0;

    for (int i = 0; i < QuestionId.Count; i++)
    {
        var question = questions.FirstOrDefault(q => q.Id == QuestionId[i]);
        if(question!=null)
        {
            var correctAnswerIndex = question.CorrectAnswerIndex;

            if (selectedAnswers[i] == correctAnswerIndex && correctAnswerIndex != null)
            {
                correctAnswersCount++;
            }
        }

    }

    var questionss = _siteDbContext.Questions.Include(q => q.Answers).Where(q => QuestionId.Contains(q.Id)).ToList();

    ViewData["CorrectAnswersCount"] = correctAnswersCount;

    return View("Result", questionss);
}
}
}
```

Answer(Model).

```
namespace Diplom.Models
{
    Ссылок: 3
    public class Answer
    {
        Ссылок: 4
        public int Id { get; set; }
        Ссылок: 3
        public string Text { get; set; } = null!;


        Ссылок: 0
        public virtual Question? Question { get; set; }
    }
}
```

Category(Model)

```
namespace Diplom.Models
{
    Ссылок: 3
    public class Category
    {
        Ссылок: 0
        public Category()
        {
            Questions = new List<Question>();
        }
        Ссылок: 2
        public int Id { get; set; }
        Ссылок: 1
        public string Name { get; set; } = null!;
        Ссылок: 1
        public virtual ICollection<Question> Questions { get; set; }
    }
}
```

News(Model).

```
using System.Text.RegularExpressions;

namespace Diplom.Models
{
    Ссылок: 16
    public class News
    {
        Ссылок: 0
        public int Id { get; set; }
        Ссылок: 6
        public string Title { get; set; } = null!;
        Ссылок: 3
        public List<string> Content { get; set; } = null!;
        Ссылок: 4
        public DateTime DatePublished { get; set; }
        Ссылок: 2
        public string Url { get; set; } = null!;
        Ссылок: 4
        public string ImageUrl { get; set; } = null!;
    }
}
```

NewsViewModel

```
namespace Diplom.Models
{
    Ссылок: 4
    public class NewsViewModel
    {
        Ссылок: 2
        public List<News> News { get; set; } = null!;
        Ссылок: 3
        public int? CurrentPage { get; set; }
        Ссылок: 1
        public string SortBy { get; set; } = null!;
    }
}
```

Photo(Model).

```
namespace Diplom.Models
{
    Ссылок: 11
    public class Photo
    {
        Ссылок: 0
        public int Id { get; set; }
        Ссылок: 2
        public string Title { get; set; } = null!;
        Ссылок: 2
        public string Src { get; set; } = null!;
    }
}
```

PhotosViewModel.

```
namespace Diplom.Models
{
    Ссылок: 4
    public class PhotosViewModel
    {
        Ссылок: 2
        public List<Photo> Photos { get; set; } = null!;
        Ссылок: 4
        public int? CurrentPage { get; set; }
    }
}
```

Question(Model).

```
namespace Diplom.Models
{
    Ссылок: 10
    public class Question
    {
        Ссылок: 0
        public Question()
        {
            Answers = new List<Answer>();
        }
        Ссылок: 4
        public int Id { get; set; }
        Ссылок: 2
        public string Text { get; set; } = null!;
        Ссылок: 2
        public int CorrectAnswerIndex { get; set; }

        Ссылок: 7
        public virtual ICollection<Answer> Answers { get; set; }
        Ссылок: 1
        public virtual Category? Category { get; set; }
    }
}
```

QuizViewModel.

```
namespace Diplom.Models
{
    Ссылок: 6
    public class QuizViewModel
    {
        Ссылок: 7
        public List<Question>? Questions { get; set; }
        Ссылок: 3
        public List<int>? SelectedAnswers { get; set; }
        Ссылок: 5
        public string ErrorMessage { get; set; } = null!;
    }
}
```

SiteDbContext.

```
namespace Diplom.Models
{
    Ссылок: 8
    public class SiteDbContext : DbContext
    {
        Ссылок: 0
        public SiteDbContext(DbContextOptions<SiteDbContext> options) : base(options) { }

        Ссылок: 4
        public virtual DbSet<Question> Questions { get; set; }
        Ссылок: 0
        public virtual DbSet<Answer> Answers { get; set; }
        Ссылок: 1
        public virtual DbSet<Category> Categories { get; set; }

        Ссылок: 0
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseLazyLoadingProxies();
        }
    }
}
```

Home: Index.cshtml(View).

```
@model NewsViewModel
@{

}
<form asp-action="Index" method="get" class="mb-3">
    <button type="submit" name="sortBy" class="btn btn-secondary btn-sm" value="Title">Сортувати за назвою</button>
    <button type="submit" name="sortBy" class="btn btn-secondary btn-sm" value="Date">Сортувати за датою</button>
</form>
<div class="text-center" style="width: 65vw">
    @foreach (var item in Model.News)
    {

        <div class="card flex flex-row mb-3 p-2" style="width: 100%">
            <img src="@item.ImageUrl" alt="@item.Title" />
            <div class="card-body">
                <h5 class="card-title">
                    <a class="title" href="@Url.Action("Index", "News", new {url = "https://www.radiosvoboda.org/" + @item.Url})">
                        @item.Title
                    </a>
                </h5>
                <h6 class="card-subtitle mb-2 text-muted">@item.DatePublished</h6>
            </div>
        </div>
    }
    <nav>
        <ul class="pagination">
            @if (Model.CurrentPage > 0 && Model.CurrentPage != null)
            {
                <form asp-controller="Home" asp-action="previousPage" method="post">
                    <button type="submit" class="page-link" aria-label="Previous">
                        <span aria-hidden="true">Попередня сторінка</span>
                    </button>
                </form>
            }
            <form asp-controller="Home" asp-action="NextPage" method="post">
                <button type="submit" class="page-link" aria-label="Next">
                    <span aria-hidden="true">Наступна сторінка</span>
                </button>
            </form>
        </ul>
    </nav>
</div>
```

News: Index.cshtml(View).

```
@model News

<link rel="stylesheet" href="~/css/news.css" asp-append-version="true" />
<div>
    <h1>@Model.Title</h1>
    <img src="@Model.ImageUrl"/>
    <hr/>
    @foreach(var content in @Model.Content)
    {
        <p class="content">@content</p>
    }
</div>
```

Photo: Index.cshtml(View).

```
@model PhotosViewModel
@{
    int? currentPage = Model.CurrentPage + 1;
}
<link rel="stylesheet" href="~/css/photos.css" asp-append-version="true" />
<div class="text-center flex flex-row">
    @foreach (var item in Model.Photos)
    {
        <div class="card mb-3 m-3" style="width: 18rem;">
            <img src="@item.Src" alt="Alternate Text" />
            <div class="card-body">
                <p class="card-text">@item.Title</p>
            </div>
        </div>
    }


</div>
<nav class="navButtons">
    <ul class="pagination">
        @if (Model.CurrentPage > 0 && Model.CurrentPage != null)
        {
            <li class="page-item">
                <a class="page-link" asp-controller="Photo" asp-action="PreviousPage" aria-label="Previous">
                    <span aria-hidden="true">&laquo;</span>
                </a>
            </li>
        }
        <li class="page-item">
            <a class="page-link">
                <span aria-hidden="true">@currentPage</span>
            </a>
        </li>
        <li class="page-item">
            <a class="page-link" asp-controller="Photo" asp-action="NextPage" aria-label="Next">
                <span aria-hidden="true">&raquo;</span>
            </a>
        </li>



    </ul>
</nav>
```

Shared: _Layout.cshtml(View).

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - UkraineWarInfo</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
    <link rel="stylesheet" href="~/Diplom.styles.css" asp-append-version="true" />
    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css" rel="stylesheet">
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm box-shadow" style="padding: 0px;">
            <div class="container-fluid">
                <a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index" onclick="SetFirstPage()">UkraineWarInfo</a>
                <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse" aria-controls="navbarSupportedContent"
                        aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
                    <ul class="navbar-nav flex-grow-1">
                        <li class="nav-item">
                            <a class="nav-link active" asp-area="" asp-controller="Home" asp-action="Index">Новини</a>
                        </li>

                        <li class="nav-item">
                            <a class="nav-link" asp-area="" asp-controller="Photo" asp-action="Index">Фото</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" asp-area="" asp-controller="Statistics" asp-action="Index">Статистика</a>
                        </li>

                        <li class="nav-item">
                            <a class="nav-link" asp-area="" asp-controller="Test" asp-action="Medicine" asp-route-category="Medicine">Тести про медицину</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" asp-area="" asp-controller="Test" asp-action="History" asp-route-category="History">Тести про війну в Україні</a>
                        </li>
                    </ul>
                </div>
            </div>
        </nav>
    </header>
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
```

```
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>

    <footer class="row row-cols-1 row-cols-sm-2 row-cols-md-5 py-5 p-2 border-top mt-5">
        <div class="col mb-3">
            <a href="/" class="d-flex align-items-center mb-3 link-dark text-decoration-none">
                <svg class="bi me-2" width="40" height="32"><use xlink:href="#bootstrap"></use></svg>
            </a>
            <p class="text-muted m-3">© UkraineWarInfo, 2024</p>
        </div>

        <div class="col mb-3">
        </div>

        <div class="navList text-center">
            <h5>Сторінки</h5>
            <ul class="navbar-nav">
                <li class="nav-item">
                    <a class="nav-link text-muted" asp-area="" asp-controller="Home" asp-action="Index" onclick="SetFirstPage()">Новини</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link text-muted" asp-area="" asp-controller="Statistics" asp-action="Index">Статистика</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link text-muted" asp-area="" asp-controller="Photo" asp-action="Index">Фото</a>
                </li>
            </ul>
        </div>

        <div class="col mb-3 text-center">
            <h5>Контакти</h5>
            <ul class="nav flex-column">
                <li class="nav-item mb-2 text-muted">+38(066)589-45-32</li>
                <li class="nav-item mb-2 text-muted">ukrainewarinfo@gmail.com</li>
            </ul>
        </div>

        <div class="col mb-3 text-center">
            <h5>Соціальні мережі</h5>
            <ul class="nav flex-row justify-content-center">
                <li class="nav-item mb-2">
                    <a data-mdb-ripple-init class="btn btn-outline-light btn-floating m-1" href="https://www.facebook.com/" role="button">
                        <i class="fab fa-facebook-f"></i>
                    </a>

                </li>
```

```
                </li>
                <li class="nav-item">
                    <a class="nav-link text-muted" asp-area="" asp-controller="Photo" asp-action="Index">Фото</a>
                </li>
            </ul>
        </div>

        <div class="col mb-3 text-center">
            <h5>Контакти</h5>
            <ul class="nav flex-column">
                <li class="nav-item mb-2 text-muted">+38(066)589-45-32</li>
                <li class="nav-item mb-2 text-muted">ukrainewarinfo@gmail.com</li>
            </ul>
        </div>

        <div class="col mb-3 text-center">
            <h5>Соціальні мережі</h5>
            <ul class="nav flex-row justify-content-center">
                <li class="nav-item mb-2">
                    <a data-mdb-ripple-init class="btn btn-outline-light btn-floating m-1" href="https://www.facebook.com/" role="button">
                        <i class="fab fa-facebook-f"></i>
                    </a>

                </li>
                <li class="nav-item mb-2">
                    <a data-mdb-ripple-init class="btn btn-outline-light btn-floating m-1" href="https://www.instagram.com/" role="button">
                        <i class="fab fa-instagram"></i>
                    </a>
                </li>
                <li class="nav-item mb-2">
                    <a data-mdb-ripple-init class="btn btn-outline-light btn-floating m-1" href="#!" role="button">
                        <i class="fab fa-twitter"></i>
                    </a>
                </li>

            </ul>
        </div>

    </footer>

    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>

    @await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```

Statistics: Index.cshtml

```
<script src="/js/warStatistic.js"></script>
<link rel="stylesheet" href="~/css/statistics.css" asp-append-version="true" />

<div>
    <h1 class="mb-5">Втрати противника на сьогодні(@DateTime.Now.Day.@DateTime.Now.Month.ToString("00"))</h1>
    <div class="wrapper mt-5" id="myDiv">
        <div>
            <img class="icon" src="/images/soldier.png" alt="units" />
            <div class="grid-item" id="personalUnits">
            </div>
        </div>
        <div>
            <img class="icon" src="/images/tank.png" alt="tanks" />
            <div class="grid-item" id="tanks">
            </div>
        </div>
        <div>
            <img class="icon" src="/images/btr.png" alt="armouredFightingVehicles" />
            <div class="grid-item" id="armouredFightingVehicles">
            </div>
        </div>
        <div>
            <img class="icon" src="/images/artillery.png" alt="artillery" />
            <div class="grid-item" id="artillerySystems">
            </div>
        </div>
        <div>
            <img class="icon" src="/images/missiles.png" alt="missiles" />
            <div class="grid-item" id="mlrs">
            </div>
        </div>
        <div>
            <img class="icon" src="/images/pvo.png" alt="pvo" />
            <div class="grid-item" id="warfareSystems">
            </div>
        </div>
        <div>
            <img class="icon" src="/images/plane.png" alt="planes" />
            <div class="grid-item" id="planes">
            </div>
        </div>
        <div>
            <img class="icon" src="/images/helicopter.png" alt="helicopter" />
            <div class="grid-item" id="helicopters">
            </div>
        </div>
        <div>
            <img class="icon" src="/images/car.png" alt="car" />
```

```
        </div>
        <div>
            <img class="icon" src="/images/pvo.png" alt="pvo" />
            <div class="grid-item" id="warfareSystems">
            </div>
        </div>
        <div>
            <img class="icon" src="/images/plane.png" alt="planes" />
            <div class="grid-item" id="planes">
            </div>
        </div>
        <div>
            <img class="icon" src="/images/helicopter.png" alt="helicopter" />
            <div class="grid-item" id="helicopters">
            </div>
        </div>
        <div>
            <img class="icon" src="/images/car.png" alt="car" />
            <div class="grid-item" id="vehiclesFuelTanks">
            </div>
        </div>
        <div>
            <img class="icon" src="/images/ship.png" alt="ship" />
            <div class="grid-item" id="warshipsCutters">
            </div>
        </div>
        <div>
            <img class="icon" src="/images/rocket.png" alt="rocket" />
            <div class="grid-item" id="cruiseMissiles">
            </div>
        </div>
        <div>
            <img class="icon" src="/images/uav.png" alt="uav" />
            <div class="grid-item" id="uavSystems">
            </div>
        </div>
        <div>
            <img class="icon" src="/images/spec-car.png" alt="spec-car" />
            <div class="grid-item" id="specialMilitaryEquip">
            </div>
        </div>
        <div>
            <img class="icon" src="/images/submarine.png" alt="submarine" />
            <div class="grid-item" id="submarines">
            </div>
        </div>
    </div>
</div>
```

Test: Questions.cshtml

```
@model QuizViewModel
@{
    ViewData["Title"] = "Test Questions";
    var questionTitle = ViewData["QuestionTitle"];

}
<link rel="stylesheet" href="~/css/test.css" asp-append-version="true" />

<h1>Питання про @questionTitle</h1>

@if (!string.IsNullOrEmpty(Model.ErrorMessage))
{
    <div class="alert alert-danger">@Model.ErrorMessage</div>
}

<form method="post" asp-action="CheckAnswers">
    @for (var i = 0; i < Model.Questions.Count; i++)
    {
        <div class="question mb-2 p-2">
            <h5 style="font-weight: bold; color: black;">@Model.Questions[i].Text</h5>
            <input type="hidden" name="QuestionId[@i]" value="@Model.Questions[i].Id"/>
            @foreach (var answer in Model.Questions[i].Answers)
            {
                <div class="answer">
                    <input type="radio" name="selectedAnswers[@i]" value="@answer.Id" /> @answer.Text
                </div>
            }
        </div>
        <hr/>
    }
    <button type="submit">Завершити</button>
</form>
```

Test: Result.cshtml

```
@model List<Question>
@{
    var countCorrectAnswers = (int?)ViewData["CorrectAnswersCount"];
}

<link rel="stylesheet" href="~/css/result.css" asp-append-version="true" />

<div class="result">
    <h1>Кількість правильних відповідей: @countCorrectAnswers</h1>
    @for (var i = 0; i < Model.Count; i++)
    {
        <div class="question mb-2 p-2">
            <h5 style="font-weight: bold; color: black;">@Model[i].Text</h5>
            @foreach (var answer in Model[i].Answers)
            {
                <div class="answer">
                    @if(answer.Id == Model[i].CorrectAnswerIndex)
                    {
                        <p type="radio" name="selectedAnswers[@i]" value="@answer.Id" style="background-color: greenyellow;"> @answer.Text</p>
                    }
                    else
                    {
                        <p type="radio" name="selectedAnswers[@i]" value="@answer.Id"> @answer.Text</p>
                    }
                </div>
            }
        </div>
        <hr />
    }
</div>
```