МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ЕНЕРГЕТИЦІ

КУРСОВА РОБОТА

з дисципліни: «Основи Веб-програмування»

на тему: «Додаток для новин»

Керівник:
Недашківський Олексій Леонідович

«Допущено до захисту»
_____
(особистий підпис керівника)

«___» _____ 20__ р.

Захищено з оцінкою
_____

(оцінка)

Виконав:
Оніщук Максим Іванович
студент 2 курсу
групи ТВ-22

_____
(особистий підпис виконавця)

«___» _____ 20__ р.

Київ – 2024

АНОТАЦІЯ

У цій курсовій роботі представлено процес розробки веб-сайту для перегляду новин. Основна увага приділяється визначенню вимог до функціоналу та дизайну сучасного новинного веб-сайту. Описано етапи створення сайту, включаючи вибір платформи та реалізацію мультимедійного контенту. Для додатку було створено достатньо зрозумілий користувацький інтерфейс, що забезпечею дуже легку навігацію у додатку

Результати цієї роботи можуть бути корисними для розробників веб-сайтів та спеціалістів з медіа, надаючи практичні рекомендації та інструменти для створення ефективного новинного ресурсу.

ANNOTATION

This term paper presents the process of developing a news website. The focus is on defining the functional and design requirements of a modern news website. The stages of site creation are described, including the choice of a platform and the implementation of multimedia content. A sufficiently clear user interface was created for the application, which ensures very easy navigation in the application

The results of this work can be useful for website developers and media professionals, providing practical guidelines and tools for creating an effective news resource.

ЗМІСТ

ВСТУП

У світі, де швидкий доступ до інформації стає все важливішим, веб-сайти новин є необхідним інструментом для отримання актуальних новин з різних джерел. Мета цього проекту полягає у створенні простого веб-сайту для перегляду новин під назвою «World shortcut», який розробляється з використанням фреймворку Flask. Вибір Flask зумовлений його легкістю, гнучкістю та зручністю для швидкого прототипування веб-додатків.

У даній роботі описано процес розробки веб-сайту від початкового етапу налаштування середовища до фінальної версії сайту. Особлива увага приділяється основним функціональним можливостям, таким як відображення контенту. Також розглядаються базові принципи дизайну, щоб забезпечити зручність та привабливість для користувачів.

Ця робота спрямована на продемонстрування можливостей фреймворку Flask у створенні ефективного та зручного новинного ресурсу. Крім того, вона надає практичні навички, які можуть бути корисними для початківців у веб-розробці, охоплюючи використання HTML та CSS для створення вигляду та оформлення веб-сайту.

1. ЗАГАЛЬНИЙ ОПИС ВЕБ ДОДАТКУ

1.1 Актуальність теми

Новина — це те важливе, що змінилося (чи не змінилось, хоча мусило), і ця зміна вплине на життя аудиторії. Інший варіант — те, що відбулося всупереч очікуванням, і це дивує аудиторію. Найскладніше в роботі новинаря — відокремити новини від не-новин, тобто не важливих для аудиторії фактів та повідомлень. Не-новини намагаються пропхати у стрічку піарники, маскуючи рекламу під справжні події. Видобувати новини з цього потоку — кропітка праця золотошукача, який вимиває крупинки золота з купи піску. Водночас ця робота вимагає постійного напруження уваги та інтуїції, яка в поєднанні зі знанням контексту дозволяє побачити неочевидну новину.[2]

Тема веб-сайтів для новин є дуже актуальною в сучасному світі з кількох причин. По-перше, інтернет стає основним джерелом новин для багатьох людей, завдяки швидкому доступу до актуальної інформації. По-друге, веб-сайти новин пропонують мультимедійний контент, включаючи відео та інфографіку, що робить новини більш доступними та зрозумілими.

Персоналізація контенту дозволяє користувачам отримувати новини, що відповідають їхнім інтересам. Крім того, адаптація до мобільних платформ забезпечує зручність отримання новин у будь-який час і місці.

Веб-сайти новин також суттєво впливають на традиційні медіа, пропонуючи інформацію швидше та в зручнішій формі. Вони відіграють важливу роль у формуванні громадської думки та політичних процесів, а також є важливими гравцями на ринку реклами завдяки великій аудиторії.

Таким чином, веб-сайти новин є важливим елементом сучасного інформаційного середовища, що постійно розвивається та адаптується до нових технологій та потреб аудиторії.

1.2 Загальний опис проекту

Для користування написаним мною веб додатком потрібно всього на всього зайти на головну сторінку. Користувач матиме можливість для вибору тематики новин, а також відфільтрувати по даті публікації. На наступному зображенні буде наведено діаграму варіантів використання для користувача:

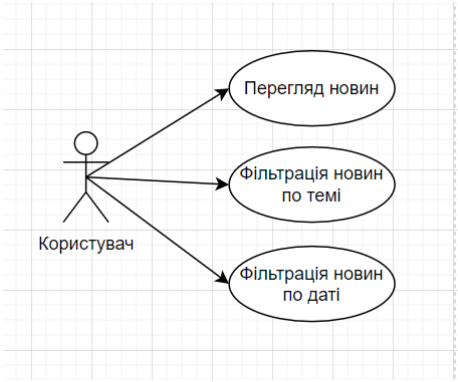

Рисунок 1.1 Діаграма варіантів використання

1.3 Інструменти розробки

Давайте розглянемо детальніше кожен інструмент та їхню роль у розробці нашого веб-сайту новин:

1. **Flask**:
    - Flask - це легкий мікрофреймворк для веб-розробки на Python. Він забезпечує базовий, але потужний набір інструментів для створення веб-додатків.
    - Flask дозволяє створювати маршрути (routes), що визначають, як додаток реагує на різні запити від користувачів.
    - Використовуючи Flask, можна легко підключити різноманітні бібліотеки та розширення для розширення функціональності додатку.
2. **HTML (HyperText Markup Language)**:
    - HTML використовується для створення структури та змісту веб-сторінок. Він визначає, які елементи (текст, зображення, відео, тощо) присутні на сторінці та їхню взаєморозташованість.

- За допомогою HTML можна створювати різноманітні елементи, такі як заголовки, абзаци, списки, таблиці тощо.

3. **CSS (Cascading Style Sheets)**:
    - CSS використовується для оформлення веб-сторінок та надання їм стилів та вигляду. Він визначає зовнішній вигляд елементів, таких як кольори, шрифти, розміри, відступи, рамки тощо.
    - CSS дозволяє розділити структуру сторінки (визначену HTML) від її вигляду та стилів, що полегшує підтримку та редагування коду.
4. **SQLite**:
    - SQLite - це легковага вбудовувана реляційна база даних, яка широко використовується в різних програмах та веб-додатках.
    - Вона забезпечує простий та зручний спосіб зберігання та організації даних без необхідності налаштовувати сервери баз даних.
    - SQLite може бути ідеальним вибором для невеликих проектів або тестування веб-сайтів, оскільки не потребує складних процедур налаштування та має низький рівень обсягу.
    - Для розробки веб-сайту новин SQLite може бути використана для зберігання та керування інформацією про новини, їх заголовками, змістом та датами публікацій.

Окрім того середовищем розробки нашого застосунку слугував PyCharm від JetBrains. PyCharm - найрозумніший Python IDE з повним набором інструментів для ефективної розробки Python. Випускається в двох версіях - безкоштовна версія PyCharm Community Edition і підтримка більшого набору функцій PyCharm Professional Edition. PyCharm виконує перевірку коду на льоту, автозаповнення, в тому числі на основі інформації, отриманої при виконанні коду, навігації по коду, забезпечує безліч рефакторингів.[1]

А для ролі системи управління базами даних було обрано також середовище від JetBrains DataGrip.

2. ОПИС СИСТЕМНОЇ РЕАЛІЗАЦІЇ

Оскільки весь процес розробки курсової роботи лежав на моїх плечах, ми розпочнемо зі створення бекенду. Це включає в себе створення серверної частини веб-додатка, яка буде відповідати за обробку запитів користувачів, забезпечення взаємодії з базою даних та надання необхідної інформації для фронтенду. Для досягнення цих цілей я використовував різні технології та інструменти, такі як Python для написання серверної логіки, Flask для реалізації веб-сервера, SQLAlchemy для роботи з базою даних та забезпечення її взаємодії з бекендом. Крім цього, важливо було забезпечити безпеку даних, ефективне управління ресурсами сервера та обробку помилок для стабільної роботи додатка.

2.1 Робота з базою даних

Наша база, реалізована на SQLite, має всього лише одну таблицю під назвою news, яка містить наступні поля:?

- NewsID – це основний ключ об’єкта
- Theme – Тематика новини (Наприклад спорт, бізнес та наука)
- Title – Заголовок новини
- Text – текст самої статті
- MediaPath – шлях до медіа файлу, зокрема певного зображення
- Date – Дата публікації новини

На наступному малюнку показано саму команду створення нашої таблиці:

```
CREATE TABLE news(
    NewsID CHAR(10) PRIMARY KEY,
    Theme VARCHAR(20) NOT NULL,
    Title VARCHAR(100) NOT NULL,
    Text TEXT NOT NULL,
    MediaPath VARCHAR(100),
    Date DATETIME DEFAULT CURRENT_TIMESTAMP NOT NULL
);
```

Рисунок 2.1.1 – Створення таблиці новин

Далі роботи з базами даних на пайтоні було використано sqlalchemy. SQLAlchemy — це популярна бібліотека на Python для роботи з базами даних. Вона надає потужний і гнучкий інструментарій для ORM (Object-Relational Mapping), що дозволяє взаємодіяти з базами даних за допомогою об'єктів Python замість написання SQL-запитів.

Для роботи з нею спочатку треба створити об'єкт “engine”, який представляє підключення до бази даних SQLite під назвою “BaseNews” на локальному хості.

```
engine = create_engine('sqlite:///BaseNews.db')
```

Рисунок 2.1.2 – Створення підключення до БД

Далі ми створюємо об’єкт “MetaData”, який зберігає інформацію про структуру таблиць у базі даних:

```
metadata = MetaData()
```

Рисунок 2.1.3 – Створення “MetaData”

Наступним кроком буде створення фабрики сесій, зв’язаної з вказаним підключенням до бази даних:

```
DBSession = sessionmaker(bind=engine)
```

Рисунок 2.1.4 – Створення фабрики сесій

Після цього створюється безпосередньо сесія “session”. Сесія використовується для виконання запитів і взаємодії з базою даних:

```
session = DBSession()
```

Рисунок 2.1.5 – Створення сесії

“Base = declarative_base()” Створює базовий клас для всіх моделей (таблиць) у базі даних. Клас “Base” використовується для визначення таблиць як Python-класів:

```
Base = declarative_base()
```

Рисунок 2.1.6 – Створення базового класу

Наступний код використовує SQLAlchemy для визначення моделі таблиці

“news” у базі даних SQLite і створення цієї таблиці, якщо вона ще не існує. Далі визначаються стовпці таблиці: NewsID (цілочисловий первинний ключ), Theme (тема новини), Title (заголовок новини), Text (текст новини), MediaPath (шлях до медіафайлу) та Date (дата публікації з індексом).

```
class News(Base):
    __tablename__ = 'news'

    NewsID = Column(Integer, primary_key=True)
    Theme = Column(String)
    Title = Column(String)
    Text = Column(String)
    MediaPath = Column(String)
    Date = Column(Date, index=True)
```

Рисунок 2.1.7 – Створення класу для роботи з таблицею

2.2 Обробка запитів та маршрутизація

Далі ми розглянемо головний модуль всього проекту – app.py. Даний файл містить в собі функції для обробки запитів POST та GET та різноманітні маршрути. На наступному рисунку зображено початкову функцію “hello_world”, яка обробляє запити які приходять на url “/templates/main.html” та “/”:

```
@app.route( rule: '/templates/main.html', methods=['POST', 'GET'])
@app.route( rule: '/', methods=['POST', 'GET'])
def hello_world():
    if request.method == 'POST':
        start_date = request.form['start_date']
        end_date = request.form['end_date']

        news_item = session.query(News).filter(and_(News.Date >= start_date,  News.Date <= end_date)).all()

        return render_template( template_name_or_list: "main.html", news=news_item)
    news_item = session.query(News).all()
    return render_template( template_name_or_list: "main.html", news=news_item)
```

Рисунок 2.2.1 – Початкова функція

Головна сторінка яка зустрічає користувача який зайшов на наш сайт, яка містить новини усіх тематик, не фільтруючи ніяк дані.

Окрім того функція перевіряє надсилання запиту POST, отримуючи з якого завдяки “request.form” отримує дані для фільтрації по даті і завдяки session.query фільтрує дані за датою та виводить сторінку з цими даними.

На наступному зображенні показано функцію для обробки запитів які надходять на адресу «/templates/economics» схоже як і минула функція Відмінність полягає в тому що тут виводить сторінка тільки для новин за темою «Економіка», так само і для інших:

Рисунок 2.2.2 – Обробка сторінки новин економіки

На наступному малюнку наведено досить просту функцію, яка виводить лише інформацію про сайт та контактні дані:

Рисунок 2.2.3 – Сторінка з інформацією про сайт

Цей код дозволяє користувачам переглядати детальну інформацію про окрему новину, звертаючись до URL з ідентифікатором новини. Якщо новина існує, вона відображається на окремій сторінці; якщо ні, користувач отримає повідомлення про помилку 404. Це забезпечує інтуїтивно зрозумілий і зручний спосіб доступу до детальної інформації про новини на веб-сайті:

Рисунок 2.2.4 – Сторінка з певною новиною

2.3 Розробка інтерфейсу користувача

Загалом у нашій файловій системі міститься один батьківський блок для інший сторінок з назвою “navigation.html”. Даний файл описує стилі для багатьох елементів нашого сайту, містить навігаційне меню, банер та футер який містить знизу сторінки. На наступному рисунку зображено верхню частину сторінки у вигляді меню та логотипу з назвою нашого веб-сайту. Даний код створює шапку веб-сторінки, яка містить банер із логотипом і назвою сайту "World shortcut", що веде на головну сторінку. Нижче розміщене навігаційне меню з посиланнями на різні розділи новин: Головна, Політика, Економіка, Спорт, Бізнес, Наука та Культура. Елементи меню організовані в список, а після списку розташована горизонтальна лінія для візуального розділення.:

```
<body>
    <header class="header">
        <div id="banner" class="banner">
            <div class="content-banner">
                <a href="../templates/main.html">
                    <img src="../static/logo.jpg" alt="Логотип" class="nav-logo">  World shortcut
                </a>
            </div>
        </div>
        <nav class="navbar">
            <ul class="nav-list">
                <li class="nav-item"><a href="../templates/main.html">Головна</a></li>
                <li class="nav-item"><a href="../templates/politics_news.html">Політика</a></li>
                <li class="nav-item"><a href="../templates/economics_news.html">Економіка</a></li>
                <li class="nav-item"><a href="../templates/sports_news.html">Спорт</a></li>
                <li class="nav-item"><a href="../templates/business_news.html">Бізнес</a></li>
                <li class="nav-item"><a href="../templates/science_news.html">Наука</a></li>
                <li class="nav-item"><a href="../templates/culture_news.html">Культура</a></li>
            </ul>
            <hr>
        </nav>
    </header>
```

Рисунок 2.3.1 – Верхнє меню сторінки

На наступному зображені буде показано сам вигляд того меню та банера нашого сайту:

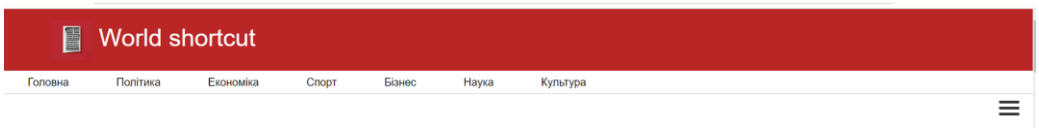

Рисунок 2.3.2 – Хедер сторінки

Код наведений нижче створює нижній колонтитул веб-сторінки, що складається з двох основних частин. Перша частина містить навігаційне меню з посиланнями на сторінки "Головна" та "Про нас, контакти", організоване у

вигляді списку. Друга частина включає інформаційний блок з текстом про авторські права: "© 2024 Сайт новин. Всі права захищені.":

```
<footer class="footer">
    <div class="footer-container">
        <ul class="footer-nav">
            <li><a href="../templates/main.html">Головна</a></li>
            <li><a href="../templates/about-us.html">Про нас, контакти</a></li>
        </ul>
    </div>
    <div class="footer-info">
        <p>© 2024 Сайт новин. Всі права захищені.</p>
    </div>
</footer>
```

Рисунок 2.3.3 – Код нижньої частини сторінки

Відповідно, як вона виглядатиме:

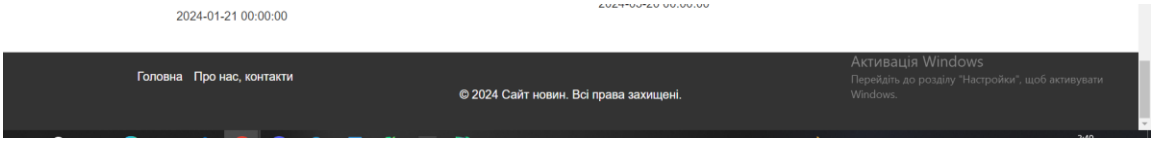

Рисунок 2.3.4 – нижня частина сторінки

Після натискання на кнопку «Про нас, контакти», нас переправить на сторінку “about-us.html”, де буде наступний текст:

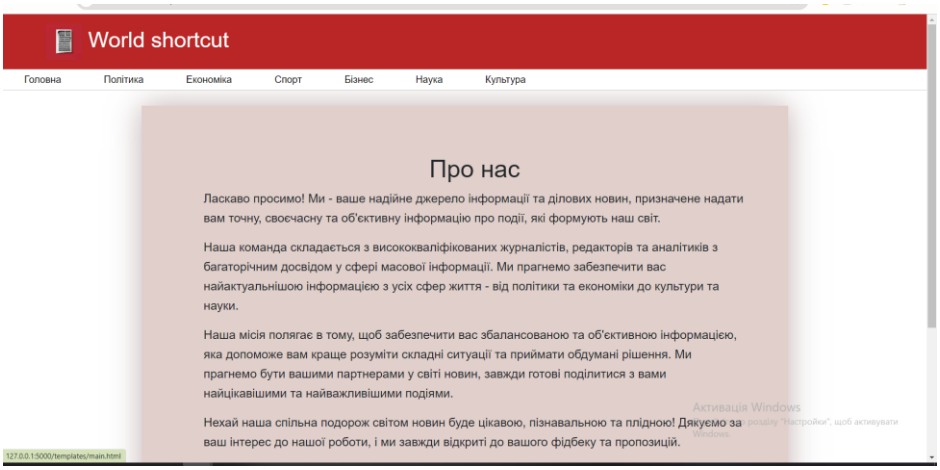

Рисунок 2.3.5

Окрім цього на сторінці кожної теми, в тому числі головній міститься праве випадаюче меню яке містить поля для фільтрації новин по датах. На рисунку нижче зображено html код даного меню. Ця форма забезпечує користувачам можливість фільтрувати новини за вибраними датами.

Користувачі можуть вибрати початкову і кінцеву дати, і натиснути кнопку "Пошук", щоб відправити ці дати на сервер. На сервері, ймовірно, є логіка для обробки цих даних і фільтрації новин відповідно до вибраного діапазону дат:

```
<form method="post">
    <input id="menu__toggle" type="checkbox">
    <label class="menu_btn" for="menu__toggle">
        <span></span>
    </label>
    <ul class="menu__box">
        <li><p class="menu__title">Фільтр за датою</p></li>
        <li><hr></li>
        <li><label for="start_date" class="menu__item">Початкова дата:</label></li>
        <li><input type="date" id="start_date" name="start_date" class="menu_input"></li>
        <li><label for="end_date" class="menu__item">Кінцева дата:</label></li>
        <li><input type="date" id="end_date" name="end_date" class="menu_input"></li>
        <li><hr></li>
        <li><button class="menu_submit">Пошук</button></li>
        </ul>
</form>
```

Рисунок 2.3.6 – html код випадаючого меню

Окрім випадаючого меню кожна сторінка має вивід списку новин. Цей фрагмент HTML-шаблону використовує Jinja2 для відображення списку новин у двох стовпцях. Кожен елемент новини відображається як посилання, яке веде до детальної сторінки новини:

```
<div class="news-container" style="...">
    <div class="news-list">
        <ul>
            {% for news_item in news %}
                {% if loop.index0 % 2 == 0 %}
            <li class="news-item">
                <a href="{{ url_for('news_detail', news_id=news_item.NewsID) }}">
                    <div class="image-container">
                        <img src="{{ news_item.MediaPath }}" alt="Image">
                    </div>
                    <h4>{{ news_item.Title }}</h4>
                    <p>{{ news_item.Date }}</p>
                </a>
            </li>
                {% endif %}
            {% endfor %}
        </ul>
```

Рисунок 2.3.7 – дівий стовпчик списку

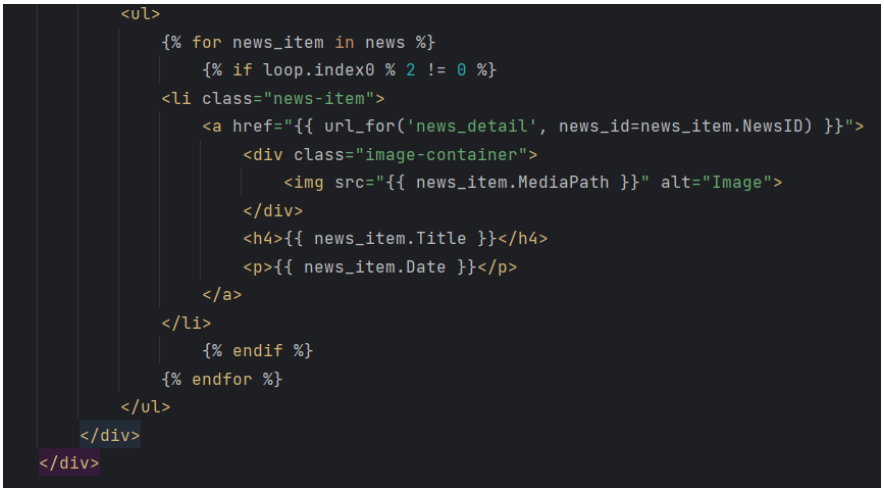

Рисунок 2.3.8 – правий стовпчик списку

У ході розробки веб-додатку для новин було створено зручний та функціональний веб-інтерфейс, який забезпечує ефективну взаємодію користувачів із сайтом. Інтерфейс включає в себе основні сторінки, такі як головна сторінка, сторінки новин за категоріями, сторінка з детальною інформацією про окремі новини, а також сторінка про нас і контакти. Веб-додаток використовує Flask для керування маршрутизацією та обробки запитів, HTML та CSS для структурування та стилізації контенту.

Завдяки цим технологіям вдалося створити інтуїтивно зрозумілу навігацію та приємний користувацький досвід. Користувачі можуть легко знаходити та переглядати новини за різними категоріями, фільтрувати новини за датою та отримувати детальну інформацію про окремі новини. Функціонал веб-додатку включає динамічне завантаження даних з бази даних та їх відображення у зручному вигляді.

2.4 Вигляд веб-додатку

Увійшовши в наш додаток, користувач відразу ж натрапить на головну сторінку яка містить різноманітні новини у вигляді стрічки. Для того щоб читати чи просто переглядати новини користувачу ніякої реєстрації проходити не треба:

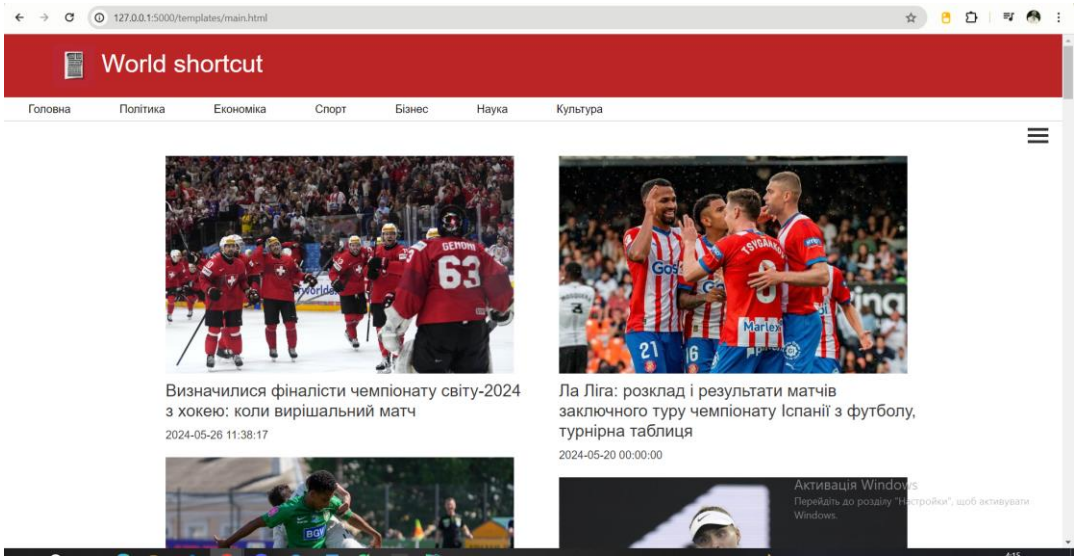

Рисунок 2.4.1 – Головна сторінка

Натиснувши на іконку в правому верхньому куті, користувач отримає доступ до випадаючого меню. У цьому меню знаходяться поля для введення початкової та кінцевої дат, що дозволяє фільтрувати новини за обраними часовими рамками безпосередньо на поточній сторінці. Завдяки цьому функціоналу, користувачі можуть легко знаходити новини, які їх цікавлять, відповідно до зазначеного періоду. Це значно підвищує зручність використання сайту та забезпечує більш персоналізований доступ до інформації. Впровадження такої функції дозволяє ефективніше обробляти великі обсяги даних і покращує загальний користувацький досвід. На рисунку нижче зображено вигляд даного меню:

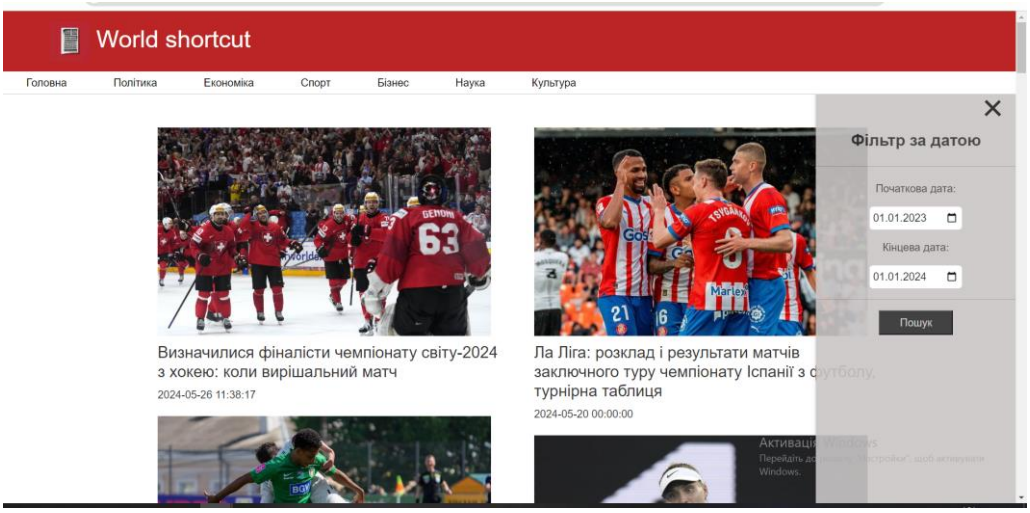

Рисунок 2.4.2 – випливаюче меню справа

При переході з однієї вкладки, наприклад, з головної на вкладку "Наука", користувача буде автоматично перенаправлено на окрему сторінку, де відображатимуться лише новини, що стосуються наукових тематик. Це забезпечує користувачам зручний доступ до інформації, яка відповідає їхнім інтересам, сприяючи кращому засвоєнню новин і підвищенню їхнього інформаційного комфорту. Така організація веб-інтерфейсу підсилює враження від використання сайту та забезпечує більш глибоке занурення користувачів у вибрані ними теми. На рисунку нижче зображено дану сторінку:

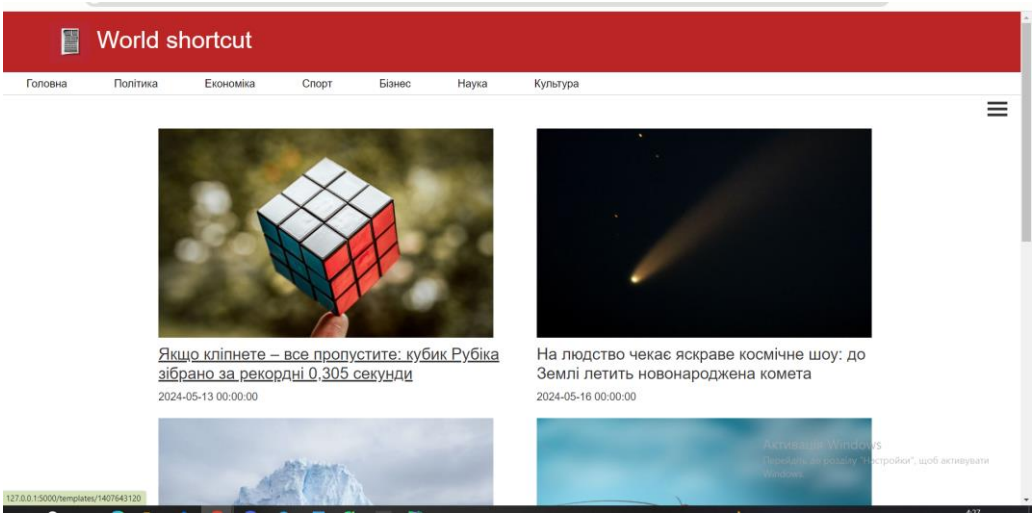

Рисунок 2.4.3 – Науковий розділ

Коли користувач натискає на будь-яку новину, він буде перенаправлений на окрему сторінку, де відображається повна інформація про обрану новину. Це включає заголовок новини, текст новини, зображення, а також дату публікації:

Рисунок 2.4.3 – Сторінка окремої новини

2.5 Написання тестів

Тести - це невід'ємна частина розробки програмного забезпечення, що дозволяє перевірити правильність роботи програми та впевнитися, що вона виконує очікувані функції. У веб-розробці тести зазвичай використовуються для перевірки різних аспектів веб-додатків, таких як коректність відображення сторінок, правильність роботи функціоналу та взаємодії з користувачем. Написання тестів дозволяє забезпечити стабільність та надійність веб-додатку, а також полегшує процес виявлення та виправлення помилок під час розробки.

Наведений нижче ‘setUp’ - це метод у класах тестів у фреймворку юніт-тестування, такому як unittest в Python. Цей метод викликається перед кожним запуском тесту і використовується для підготовки потрібних умов для виконання тесту. Наприклад, встановлення з'єднання з базою даних, створення тестових об'єктів або будь-які інші дії, які необхідно виконати перед кожним тестом для його коректної роботи. В нашому випадку тут створюється тестова новина:

```
class MyTestCase(unittest.TestCase):

    def setUp(self):
        self.app = app.test_client()
        self.app.testing = True

        # Create a test news item
        self.news_item = News(
            NewsID=1546666666,
            Theme="Sport",
            Title="Test News",
            Text="Test content",
            MediaPath="path/to/image",
            Date=datetime.now()
        )
        session.add(self.news_item)
        session.commit()
```

Рисунок 2.5.1 – підготовка до тестування

Але поки функція “setUp” використовується для підготовки середовища до тестування, то функція “tearDown” навпаки використовуєтьсядля очищення середовища

```
    def tearDown(self):
        # Remove the test news item
        session.delete(self.news_item)
        session.commit()
```

Рисунок 2.5.2 – очищення після тестування

“test_home_page”: Цей тест перевіряє, чи коректно працює головна сторінка нашого веб-додатку. Він відправляє GET-запит на головну сторінку та перевіряє, чи повертає вона код статусу 200 (OK). Потім тест перевіряє, чи присутній текст новини "Test News" на сторінці

```
    def test_home_page(self):
        response = self.app.get('/')
        self.assertEqual(response.status_code, second: 200)
        self.assertIn( member: b'Test News', response.data)
```

Рисунок 2.5.3 = перший тест

test_sport_news_page: Цей тест перевіряє, чи коректно працює сторінка спортивних новин. Він також відправляє GET-запит на сторінку спортивних новин та перевіряє код статусу 200 та наявність тексту новини "Test News".

```
    def test_sport_news_page(self):
        response = self.app.get('/templates/sports_news.html')
        self.assertEqual(response.status_code, second: 200)
        self.assertIn( member: b'Test News', response.data)
```

Рисунок 2.5.4 – Другий тест

test_news_detail_page: Цей тест перевіряє, чи коректно працює сторінка деталей новини. Він відправляє GET-запит на сторінку з деталями новини за її унікальним ідентифікатором та перевіряє, чи повертає вона код статусу 200 та чи присутній текст новини "Test News".

```
    def test_news_detail_page(self):
        response = self.app.get('/templates/1546666666')
        self.assertEqual(response.status_code, second: 200)
        self.assertIn( member: b'Test News', response.data)
```

Рисунок 2.5.5 – Третій тест

test_filter_news_by_date: Цей тест перевіряє функціональність фільтрації новин за датою на головній сторінці. Він надсилає POST-запит з встановленими початковою та кінцевою датами, а потім перевіряє, чи повертає сторінка код статусу 200 та чи присутній текст новини "Test News".

```
    def test_filter_news_by_date(self):
        response = self.app.post( *args: '/', data=dict(
            start_date='2024-01-01',
            end_date='2024-12-31'
        ))
        self.assertEqual(response.status_code, second: 200)
        self.assertIn( member: b'Test News', response.data)
```

Рисунок 2.5.6 – Четвертий тест

test_no_news_found: Цей тест перевіряє, чи коректно працює сторінка бізнес-новин, якщо жодної новини не знайдено. Він перевіряє, чи повертає сторінка код статусу 200 та чи відсутній текст новини "Test News".

```
    def test_no_news_found(self):
        response = self.app.get('/templates/business_news.html')
        self.assertEqual(response.status_code, second: 200)
        self.assertNotIn( member: b'Test News', response.data)
```

Рисунок 2.5.7 - П’ятий тест

test_found_about: Цей тест перевіряє, чи коректно працює сторінка "Про нас, контакти". Він перевіряє, чи повертає сторінка код статусу 200 та чи відсутній текст новини "Test News".

```
    def test_found_about(self):
        response = self.app.get('/templates/about-us.html')
        self.assertEqual(response.status_code, second: 200)
        self.assertNotIn( member: b'Test News', response.data)
```

Рисунок 2.5.8 – Шостий тест

На наступному рисунку буде зображено результати ці всіх тестів:

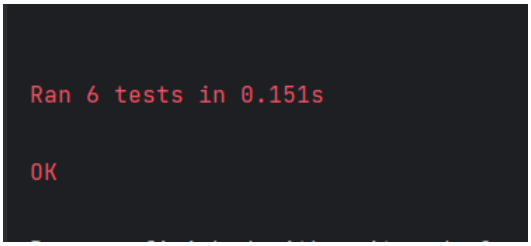

Рисунок 2.5.9 – Результат тестування

2.5 Деплой .Хостинг веб-сайту

Перш за все потрібно завантажити наш проект на GitHub. На рисунку нижче зображено репозиторій з нашим проектом:

Рисунок 2.5.1 – Репозиторій проекту

Для деплою нашого веб-сайту використовувався сервіс Render. Перш за все, щоб почати працювати з ним потрібно зайти туди через свій аккаунт GitHub. Після цього було створено файл “requirements.txt”, в якому містяться наші Python-залежності. Далі переходимо на панель керування, де натискаємо на кнопку “New” і вибираємо тип “Web Service”.

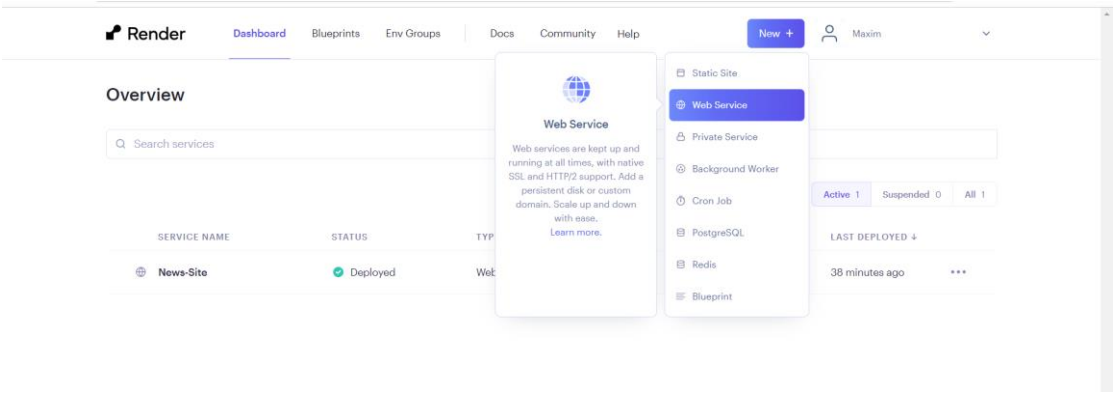

Рисунок 2.5.2 – Підготовка до деплою

З підключеного до Render нашого облікового запису GitHub вибираємо репозиторій в якому містився наш проект веб сервісу. Налаштовуємо середовище, в нашому випадку це Python, та вказуємо для запуску нашого додатку, для нас це: “python app.py”.

Для запуску нашого деплою потрібно натиснути кнопку “Manual Deploy” та вибираємо “Deploy last commit”, після чого запускається збір нашого додатку та його деплой. Під час цього можна переглядати логи деплою для перевірки статусу та усунення можливих помилок. У разі успішоного деплою ми отримаємо:

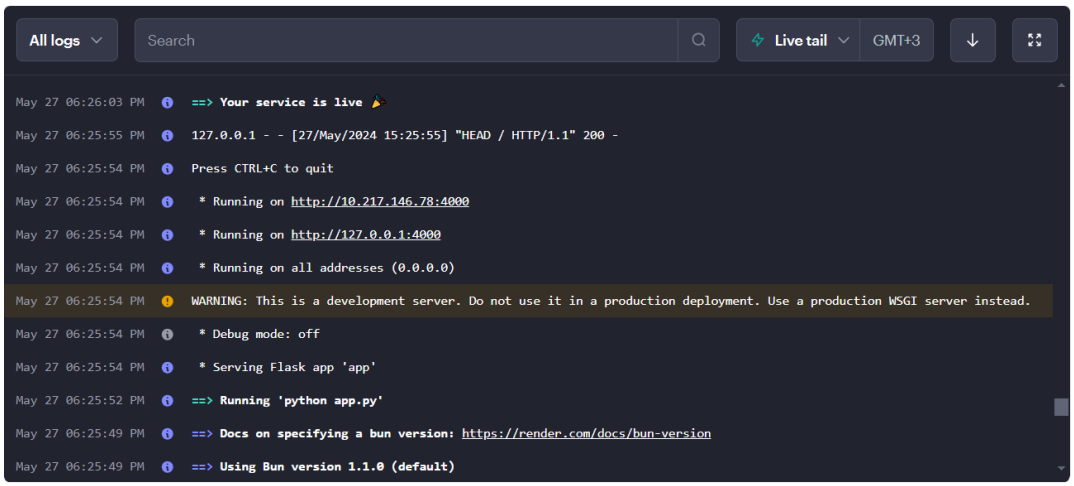

Рисунок 2.5.3 – лог успішного деплою

Після успішоного деплою сервіс надає url адресу нашого сайту та як наведено на наступному рисунку:

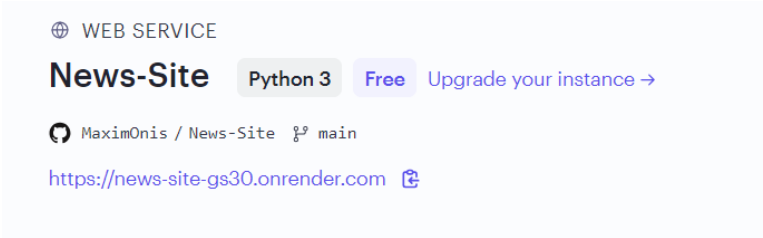

Рисунок 2.5.4 – Посилання на сайт

Перейшовши за цим посиланням, ми запевнимось в успішному запуску:

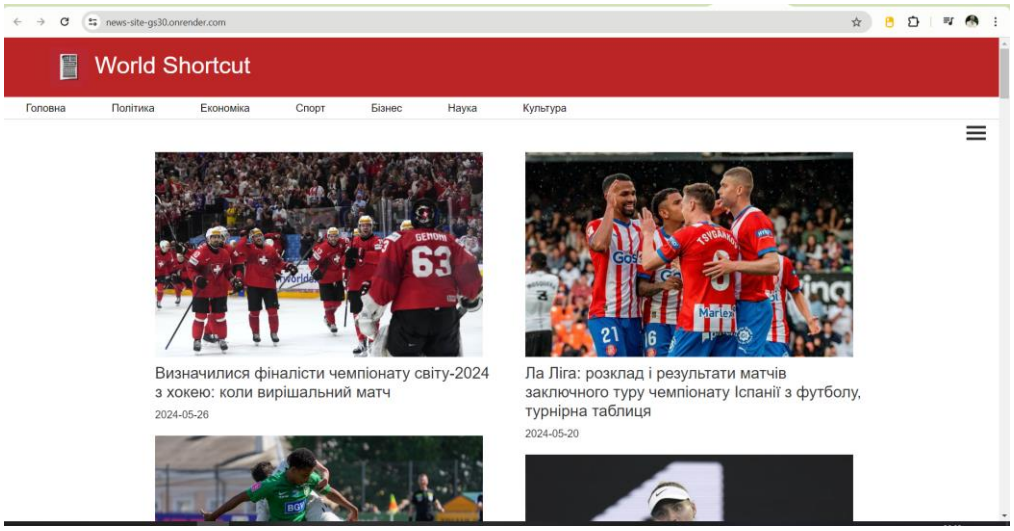

Рисунок 2.5.5 – Перевірка посилання

ВИСНОВОК

У процесі написання курсової роботи був розроблений веб-додаток для новин на Flask з використанням HTML та CSS під назвою “World shortcut”. Цей проект дозволяє користувачам зручно переглядати актуальні новини та фільтрувати їх за різними категоріями.

Веб-додаток складається з різних сторінок, таких як головна, сторінки категорій новин, сторінки деталей новин тощо. Кожна сторінка має зручний та естетично приємний дизайн, розроблений з використанням CSS, що дозволяє користувачам комфортно переглядати контент.

Завдяки використанню Flask, веб-додаток має зручну структуру та можливість швидко реагувати на запити користувачів. HTML використовується для верстки сторінок, а CSS — для оформлення їх візуальної частини, що забезпечує зручність та привабливість інтерфейсу.

В результаті розробки цього веб-додатка користувачі отримують зручний інструмент для отримання актуальних новин та оцінки їх важливості та релевантності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Itpro. [Електроний ресурс]

   Режим доступу: https://itpro.ua/product/jetbrains-pycharm/?tab=description
2. Medialab Як писати новини [Електроний ресурс]

   Режим доступу: https://medialab.online/news/novyny1

Додатки

Додаток А. Фрагмент коду, файл app.py

```
from flask import Flask, render_template, request, abort
from model import session, News, and_


app = Flask(__name__)


@app.route('/templates/about-us.html', methods=['GET'])
def about():
    return render_template('about-us.html')


@app.route('/templates/sports_news.html', methods=['POST', 'GET'])
def sport_news():
    if request.method == 'POST':
        start_date = request.form['start_date']
        end_date = request.form['end_date']

        news_items = session.query(News).filter(and_(News.Date >= start_date, News.Date <= end_date, News.Theme == "Sport")).all()

        return render_template("category_news.html", news=news_items)
    selected_news = session.query(News).filter_by(Theme='Sport').all()

    return render_template('category_news.html', news=selected_news)


@app.route('/templates/business_news.html', methods=['POST', 'GET'])
def business_news():
    if request.method == 'POST':
        start_date = request.form['start_date']
```

```
        end_date = request.form['end_date']

        news_items = session.query(News).filter(and_(News.Date >= start_date, News.Date <= end_date, News.Theme == "Business")).all()

        return render_template("category_news.html", news=news_items)
    selected_news = session.query(News).filter_by(Theme='Business').all()

    return render_template('category_news.html', news=selected_news)


@app.route('/templates/culture_news.html', methods=['POST', 'GET'])
def culture_news():
    if request.method == 'POST':
        start_date = request.form['start_date']
        end_date = request.form['end_date']

        news_items = session.query(News).filter(and_(News.Date >= start_date, News.Date <= end_date, News.Theme == "Culture")).all()

        return render_template("category_news.html", news=news_items)
    selected_news = session.query(News).filter_by(Theme='Culture').all()

    return render_template('category_news.html', news=selected_news)


@app.route('/templates/economics_news.html', methods=['POST', 'GET'])
def economics_news():
    if request.method == 'POST':
        start_date = request.form['start_date']
        end_date = request.form['end_date']

        news_items = session.query(News).filter(and_(News.Date >= start_date, News.Date <= end_date, News.Theme == "Economics")).all()

        return render_template("category_news.html", news=news_items)
```

```
    selected_news = session.query(News).filter_by(Theme='Economics').all()

    return render_template('category_news.html', news=selected_news)


@app.route('/templates/politics_news.html', methods=['POST', 'GET'])
def politics_news():
    if request.method == 'POST':
        start_date = request.form['start_date']
        end_date = request.form['end_date']

        news_items = session.query(News).filter(and_(News.Date >= start_date, News.Date <= end_date, News.Theme == "Politics")).all()

        return render_template("category_news.html", news=news_items)
    selected_news = session.query(News).filter_by(Theme='Politics').all()

    return render_template('category_news.html', news=selected_news)


@app.route('/templates/science_news.html', methods=['POST', 'GET'])
def science_news():
    if request.method == 'POST':
        start_date = request.form['start_date']
        end_date = request.form['end_date']

        news_items = session.query(News).filter(and_(News.Date >= start_date, News.Date <= end_date, News.Theme == "Science")).all()

        return render_template("category_news.html", news=news_items)
    selected_news = session.query(News).filter_by(Theme='Science').all()

    return render_template('category_news.html', news=selected_news)


@app.route('/templates/<int:news_id>', methods=['GET'])
```

```
def news_detail(news_id):
    selected_news = session.query(News).filter_by(NewsID=news_id).first()
    if selected_news is None:
        abort(404)
    return render_template('news_detail.html', news_item=selected_news)


@app.route('/templates/main.html', methods=['POST', 'GET'])
@app.route('/', methods=['POST', 'GET'])
def hello_world():
    if request.method == 'POST':
        start_date = request.form['start_date']
        end_date = request.form['end_date']

        news_item = session.query(News).filter(and_(News.Date >= start_date, News.Date <= end_date)).all()

        return render_template("main.html", news=news_item)
    news_item = session.query(News).all()
    return render_template("main.html", news=news_item)


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=4000)
```