

**Міністерство освіти і науки України
Чернівецький національний університет
імені Юрія Федьковича**

Факультет математики та інформатики
(повна назва інституту/факультету)

Кафедра математичного моделювання
(повна назва кафедри)

**Розробка web-додатку для виявлення
та аналізу фейкових новин в Інтернет
Кваліфікаційна робота
Рівень вищої освіти — другий (магістерський)**

Виконав:

студент 6 курсу, групи 601

Тритенко Юрій Миколайович

Керівник доцент, кандидат фіз.-мат. наук

Юрченко Ігор Валерійович

До захисту допущено
на засіданні кафедри
математичного моделювання
Протокол № 5 від 26 листопада 2024 р.
Зав. кафедри _____ проф. Черевко І. М.

Чернівці – 2024

АНОТАЦІЯ

Метою кваліфікаційної роботи є розробка Веб-додатку для виявлення та аналізу фейків в інтернеті. Програмна реалізація системи виконана на мові C# із використанням середовища Visual Studio та технологію Asp.Net.

Магістерська робота містить 3 розділи в яких детально описано використані технології, опис функцій програми та декілька прикладів роботи додатку.

Ключові слова: веб-додаток, C#, Asp.Net, Sql, Ml.Net.

Робота містить результати власних досліджень. Використання чужих ідей, результатів і текстів мають посилання на відповідне джерело.

ANNOTATION

The purpose of this qualification work is the development of a web application for detecting and analyzing fake content on the internet. The system's implementation is carried out using the C# programming language with the Visual Studio environment and Asp.Net technology. The master's thesis consists of three sections, which provide a detailed description of the technologies used, the program's functions, and several examples of the application in operation.

Keywords: web application, C#, Asp.Net, Sql, Ml.Net.

The work contains the results of original research. The use of others' ideas, results, and texts includes references to the corresponding sources.

_____ Ю.М Тритенко
(підпис)

ЗМІСТ

ВСТУП	6
Розділ 1	8
1.1 Аналіз об'єкту дослідження та предметної області .	8
1.1.1 Теоретичні відомості про об'єкт дослідження та предметну область	8
1.1.2 Принцип взаємодії веб-додатку	11
1.2. Постановка задачі	12
1.2.1 Вимоги до системи	12
1.2.2 Очікування від впровадження системи	14
Висновки до розділу 1	14
РОЗДІЛ 2	16
ПОБУДОВА СИСТЕМИ	16
2.1 Моделювання системи	16
2.1.1 Функції та структура системи	16
2.1.2 Опис роботи системи	17

2.2 Аналіз та вибір технологій для веб-додатку	20
Висновки до розділу 2.....	22
РОЗДІЛ 3.	24
ПРАКТИЧНА РЕАЛІЗАЦІЯ	24
3.1 Засоби розробки	24
3.2 Опис файлової структури проєкту.....	25
3.3 Опис реалізації системи	28
3.4 Аналіз отриманих результаті	34
ВИСНОВКИ	48
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	49
Додаток А. Код програми	50

ВСТУП

Метою даної кваліфікаційної роботи є розробка Веб-додатку для виявлення та аналізу фейків в інтернеті.

Мета дослідження полягає в розробці ефективного і надійного веб-додатку для виявлення та аналізу фейків в інтернеті. Така система дуже важлива в даний час через через мірне розповсюдження дезінформації в інтернеті.

Основні завдання дослідження включають аналіз предметної області, проектування веб-додатку та вибір відповідного інструментарію та методів для реалізації цієї системи. Аналіз предметної області передбачає вивчення сучасних технологій та рішень у галузі аналізу інформації, а також виявлення основних потреб і вимог користувачів. На основі цього аналізу буде проведено проектування веб-додатку, який включатиме розробку інтерфейсу користувача для зручної взаємодії з сайтом, база даних для збереження всієї потрібної інформації про користувача та серверна частина для обробки всіх вхідних даних.

Основна ідея полягає в тому, що користувач надає інформацію про новину яку він хоче перевірити. Наступним етапом буде отримання результатів правдивості цієї новини. Таким чином, веб-додаток для виявлення та аналізу фейків в інтернеті буде забезпечувати користувача інформацією чи довіряти новині чи ні.

Загалом, розробка такого веб-додатку має великий потенціал для забезпечення економії часу користувача через те що користувачеві не потрібно буде переглядати багато ресурсів для отримання інформації чи фейкова новина чи ні.

Завдання дослідження включають:

- Аналіз предметної області, дослідження потреб користувачів та визначення необхідного функціоналу для інтерфейсу користувача.
- Проектування веб-додатку, включаючи створення концептуальної, логічної та фізичної моделей бази даних для збереження інформації про користувачів та новини.

- Вибір та використання необхідних інструментів та методів для реалізації інтерфейсу користувача.
- Розробка функціоналу та інтерфейсу для реєстрації та авторизації користувачів, перегляд і.

Програмна реалізація системи виконана мовою C# за допомогою середовище програмування Visual Studio та за допомогою таких технологій як Asp.Net та Ml.Net.

У результаті виконання магістерської роботи розроблено веб-додаток для виявлення та аналізу фейків в інтернеті.

Об'єктом дослідження є веб-додаток для виявлення та аналізу фейків в інтернеті.

Предметом дослідження є технології проектування та розробки веб-додатку, методики аналізу та оцінки веб-додатків, а також використання популярних технологій для розробки.

Розділ 1

1.1 Аналіз об'єкту дослідження та предметної області

1.1.1 Теоретичні відомості про об'єкт дослідження та предметну область

Веб-Додаток - це програмне забезпечення, яке виконується на сервері та доступне користувачам через веб-браузер. Воно поєднує функціональність класичних програм із перевагами доступності через інтернет.

Зазвичай веб-додатки створюються для вирішення певних завдань, таких як обробка даних, взаємодія з базами даних, автоматизація процесів або надання інтерактивних сервісів. Вони використовують веб-технології, такі як HTML, CSS, JavaScript, а для серверної частини — мови програмування (наприклад, C#, Python, PHP) і фреймворки, як-от Asp.Net або Django.

Апаратне забезпечення - це фізичні пристрої, такі як комп'ютери, сервери, мережеві пристрої, сенсори, пристрої зберігання даних та інші обчислювальні пристрої, які використовуються для обробки і зберігання інформації.

Програмне забезпечення - це програми та системи, які встановлюються на апаратному забезпеченні для виконання конкретних функцій. Вони можуть включати операційні системи, бази даних, програми обробки даних, додатки для взаємодії з користувачем та інші спеціалізовані програми.

Бази даних - це організовані зберігальні структури, які використовуються для зберігання великого обсягу даних. Бази даних забезпечують доступ, пошук, модифікацію та управління даними, що використовуються в веб-додатку.

Мережеві зв'язки - це інфраструктура, що дозволяє передавати дані та забезпечувати зв'язок між різними компонентами системи. Мережеві зв'язки можуть включати локальні мережі (LAN), бездротові мережі (Wi-Fi, Bluetooth), глобальну мережу Інтернет та інші комунікаційні канали.

Людські ресурси - це люди, які взаємодіють з веб-додатком, включаючи адміністраторів, програмістів, користувачів та інших фахівців. Людські ресурси забезпечують конфігурацію, розробку, впровадження, підтримку та використання веб-додатку.

Ці елементи взаємодіють між собою для забезпечення збору, зберігання, обробки, передачі та використання інформації у рамках веб-додатку. Кожен з цих елементів відіграє важливу роль у функціонуванні та ефективності системи в цілому.

Для початку розробки інтерфейсу користувача потрібно ознайомитися з деякими поняттями:

UI (User Interface) - це те, як користувач взаємодіє з веб-додатком або програмним додатком. Це включає в себе всі елементи, з якими користувач може спілкуватися, такі як кнопки, меню, форми введення даних, списки, зображення та інші візуальні елементи. Інтерфейс користувача повинен бути зрозумілим, легким у використанні та забезпечувати зручну навігацію, щоб користувач міг ефективно взаємодіяти з системою.

UX (User Experience) - це враження, яке отримує користувач під час взаємодії з веб-додатком або програмним додатком. UX охоплює всі аспекти взаємодії користувача, включаючи візуальний дизайн, функціональність, легкість використання, продуктивність та задоволення, яке користувач отримує від використання системи. Головною метою UX є створення задоволення та позитивного враження у користувачів, сприяючи вирішенню їхніх потреб і досягненню поставлених цілей.

Узгоджена робота UI та UX допомагає створити зручну та задоволену користувачем взаємодію з веб-додатком або програмним додатком, покращуючи їхню використовуваність та ефективність.

Для моделювання інтерфейсу користувача ми будемо використовувати Figma.

Figma - це онлайн-інструмент для дизайну і прототипування, призначений для створення користувацького інтерфейсу (UI) для веб-сайтів, мобільних додатків і інших цифрових продуктів. Він надає

можливості спільної роботи в режимі реального часу, що дозволяє дизайнерам, розробникам та іншим зацікавленим сторонам спільно працювати над проектами, коментувати та робити зміни одночасно.

Figma має візуальний редактор, що дозволяє створювати векторні графічні елементи, інтерфейси та макети. Він надає широкий набір інструментів для малювання, редагування, вирізання, групування та організації різних шарів і компонентів дизайну. Figma також підтримує стилі, компоненти та бібліотеки, що дозволяє створювати і використовувати повторювані елементи і забезпечує їх автоматичне оновлення в усіх макетах, де вони використовуються.

Для реалізації веб-додатку ми будемо використовувати середовище розробки Visual Studio та мову програмування C# а також такі технології як Asp.Net та Ml.Net.

C# – це статично типізована мова програмування, яка працює на платформі .NET. Вона була розроблена компанією Microsoft з метою створення сучасної, експресивної та безпечної мови програмування для розробки програмного забезпечення на базі Common Language Runtime (CLR), а також для розширення можливостей розробки на різних платформах, включаючи Windows, macOS, Linux та мобільні пристрої.

C# поєднує в собі переваги об'єктно-орієнтованого та функціонального програмування, надаючи розробникам потужні інструменти для створення ефективного, масштабованого та зрозумілого коду. Вона має читабельний синтаксис, полегшує роботу з нульовими значеннями (null safety) за допомогою Nullable Reference Types та забезпечує вбудовану підтримку асинхронного програмування через `async/await`.

C# може використовуватись для розробки різноманітних застосунків, включаючи десктопні програми, веб-сервіси, ігри, хмарні рішення та мобільні додатки за допомогою Xamarin або .NET MAUI. Вона інтегрується з існуючими бібліотеками та кодом на інших мовах платформи .NET, що забезпечує гнучкість і зручність для розробників.

C# продовжує залишатися однією з найпопулярніших мов програмування завдяки своїй багатофункціональності, високій

продуктивності та широкій підтримці з боку розробників інструментів, фреймворків і спільноти.

Visual Studio — це інтегроване середовище розробки (ICP), створене компанією Microsoft. Воно надає розробникам потужний набір інструментів для створення, тестування, налагодження та розгортання програмного забезпечення для різних платформ, включаючи Windows, macOS, Linux, веб-додатки, ігри та хмарні сервіси.

Visual Studio базується на технологіях Microsoft і забезпечує широкий спектр можливостей, спеціально налаштованих для розробки на платформах .NET і C#. Воно включає в себе такі інструменти, як зручний редактор коду з підсвічуванням синтаксису, вбудований дебагер, інтеграція з Git, потужний профайлер, засоби для роботи з базами даних, редактор інтерфейсів, емулятори для тестування додатків і багато іншого.

Visual Studio також підтримує розробку багатоплатформних рішень за допомогою .NET MAUI, Xamarin, Unity та інших фреймворків, що дозволяє створювати десктопні, мобільні та веб-додатки в одному середовищі. Завдяки своїй функціональності, розширюваності та підтримці з боку Microsoft, Visual Studio є популярним вибором серед розробників програмного забезпечення різного рівня.

ASP.NET — це фреймворк для веб-розробки, створений компанією Microsoft, який дозволяє створювати динамічні веб-додатки, сервіси та API. Він є частиною платформи .NET і надає розробникам інструменти для швидкої та ефективної розробки веб-рішень з високою продуктивністю.

ML.NET — це бібліотека машинного навчання, розроблена Microsoft, яка дозволяє створювати, тренувати та впроваджувати моделі машинного навчання безпосередньо в додатках на .NET. Вона орієнтована на розробників, які працюють із C# або F#, і забезпечує простий і зрозумілий інтерфейс для інтеграції AI-рішень у ваші програми.

1.1.2 Принцип взаємодії веб-додатку

Принцип взаємодії веб-додатку в забезпеченні зручного та ефективного способу виявлення та аналізу правдивості новини. Цей принцип включає в себе різні аспекти, такі як комунікаційні протоколи, засоби перевірки новини та відображення інформації.

Перш за все, веб-додаток повинен мати можливість перегляду актуальних новин на сьогоднішній день.

Другим аспектом є інтерфейс користувача. Він є інтерактивною межею між користувачем та додатком. Інтерфейс користувача реалізовано в вигляді веб додатку для більшої зручності та мобільності.

Головна мета інтерфейсу користувача полягає в тому, щоб зробити взаємодію максимально зручною та доступною для користувача. Це може бути досягнуто шляхом простоти використання, інтуїтивно зрозумілих елементів керування та зручного представлення інформації.

Загалом, принцип взаємодії веб-додатку передбачає створення зручної та інтуїтивно зрозумілої системи, яка дозволяє користувачу легко аналізувати новини та отримувати детальний опис чому ця новина або правдива або ні.

1.2. Постановка задачі

1.2.1 Вимоги до системи

Розроблена веб-додаток складається з інтерфейсу користувача та серверної частини.

До інтерфейсу користувача ставляться такі вимоги:

- ✖ Простота використання - Інтерфейс повинен бути легким у навігації та розумінні. Користувач повинен швидко зрозуміти, як взаємодіяти з системою та здійснювати необхідні дії без заплутування.
- ✖ Інтуїтивність - Інтерфейс повинен використовувати знайомі символи, іконки та мову, щоб користувачам було зрозуміло, що робити і як керувати сайтом. Він повинен враховувати загальноприйняті стандарти та конвенції в дизайні інтерфейсу.

- ✖ Оперативність - Інтерфейс повинен бути швидким та відгукливим. Користувач повинен мати змогу швидко здійснювати дії та отримувати миттєвий зворотний зв'язок від системи.
- ✖ Інформативність - Інтерфейс повинен надавати користувачу достатньо інформації про новини, профіль користувача та історію перевірених новин. Це допоможе користувачу зрозуміти, які новини зараз популярні та якими новинами вже цікавився наш користувач.
- ✖ Сумісність з різними пристроями - Інтерфейс повинен бути сумісним з різними пристроями. Він може бути доступний через мобільні браузерери або веб-браузери.

Програмну реалізацію виконано на мові програмування C# використовуючи середовище програмування Visual Studio.

До серверної частини ставять наступні вимоги:

- ✖ Продуктивність - Серверна частина повинна забезпечувати швидке оброблення запитів користувачів, навіть при значному навантаженні. Система повинна оптимально використовувати ресурси для зменшення затримок і забезпечення стабільної роботи.

- ✖ Масштабованість - Серверна архітектура повинна підтримувати горизонтальне та вертикальне масштабування для оброблення зростаючої кількості користувачів і запитів. Це дозволить системі залишатися ефективною навіть за умов значного збільшення навантаження.

- ✖ Надійність - Серверна частина повинна мати високу відмовостійкість. У разі несправностей або збоїв система повинна автоматично відновлюватися, зберігаючи дані та продовжуючи роботу без значного впливу на користувачів.

- ✖ Безпека - Сервер повинен забезпечувати захист даних користувачів, зокрема їхніх облікових записів, історії та профілів. Це включає шифрування передавання даних, автентифікацію, авторизацію та захист від атак, таких як SQL-ін'єкції чи DDoS.

- ✱ Гнучкість - Серверна частина повинна підтримувати модульну структуру, що дозволяє легко оновлювати чи додавати нові функції без значних змін у загальній архітектурі.

- ✱ Інтеграція - Система повинна бути здатною інтегруватися з іншими сервісами та API для розширення функціональності, наприклад, для перевірки новин, роботи з базами даних чи хмарними сервісами.

1.2.2 Очікування від впровадження системи

Впровадження веб-додатку з інтуїтивно зрозумілим інтерфейсом користувача забезпечує простий спосіб перевірки новин на достовірність. Користувачі зможуть швидко зорієнтуватися в інтерфейсі, незалежно від рівня технічних знань. Завдяки чітко організованим елементам навігації та візуальним підказкам, система сприятиме мінімізації часу, витраченого на перевірку інформації. Це створює зручний інструмент для користувачів, які прагнуть швидко отримувати результати без зайвих складнощів. Архітектура системи спроектована таким чином, щоб підтримувати зростання кількості користувачів і збільшення обсягу даних. Завдяки масштабованості, система може обробляти значні навантаження без втрати продуктивності. Це відкриває можливості для розширення її функціоналу в майбутньому та забезпечує стабільну роботу навіть у пікові моменти активності користувачів. Важливим аспектом системи є високий рівень безпеки, який включає шифрування передавання даних, захист від несанкціонованого доступу та забезпечення конфіденційності даних користувачів. Система підтримує сучасні механізми автентифікації та авторизації, що гарантує збереження персональної інформації користувачів навіть у випадку підвищеної активності або атак на серверну частину.

Висновки до розділу 1

У даному розділі було наведені теоретичні відомості про веб-додаток, описано принцип роботи системи та принцип роботи клієнта з

сайтом. Також було виконано постановку задачі з описом вимог до системи та очікуваннями від її впровадження.

Після аналізу аналогів які є на ринку було прийнято рішення створити власний веб-додаток для виявлення та аналізу фейків в інтернеті. Це рішення було прийнято через малу конкурентність в даній області. Розробка власного веб-додатку допоможе значно знизити витрати часу та допоможе людям виявляти фейки в інтернеті не маючи при цьому деякого багажу знань за спиною.

Хоча наш додаток не матиме такої якості як , деякі з аналогів але вона має потенціал для подальшого розвитку. Власний додаток дозволить гнучко адаптуватися до потреб користувачів. Крім того, це надасть можливість розширювати функціональність додатку.

Для розробки веб-додатку було обрано мова програмування C#, технології Asp.Net, Ml.Net та середовище програмування Visual Studio. C# є потужною та універсальною мовою програмування, яка широко використовується для створення програмних рішень на платформі .NET. C# як мова програмування забезпечує розробника широким набором інструментів для реалізації складного функціоналу, зберігаючи зручність та простоту кодування. Це дозволяє створювати масштабовані та продуктивні рішення, що відповідають вимогам сучасних користувачів. Visual Studio надає розробнику зручне середовище для розробки, що включає інтелектуальну систему автодоповнення, відладку, інтеграцію з контролем версій та інструменти для створення інтерфейсу користувача.

РОЗДІЛ 2

ПОБУДОВА СИСТЕМИ

2.1 Моделювання системи

2.1.1 Функції та структура системи

Структура веб-додатку складається із декількох таблиць в базі даних (рис. 2.1).

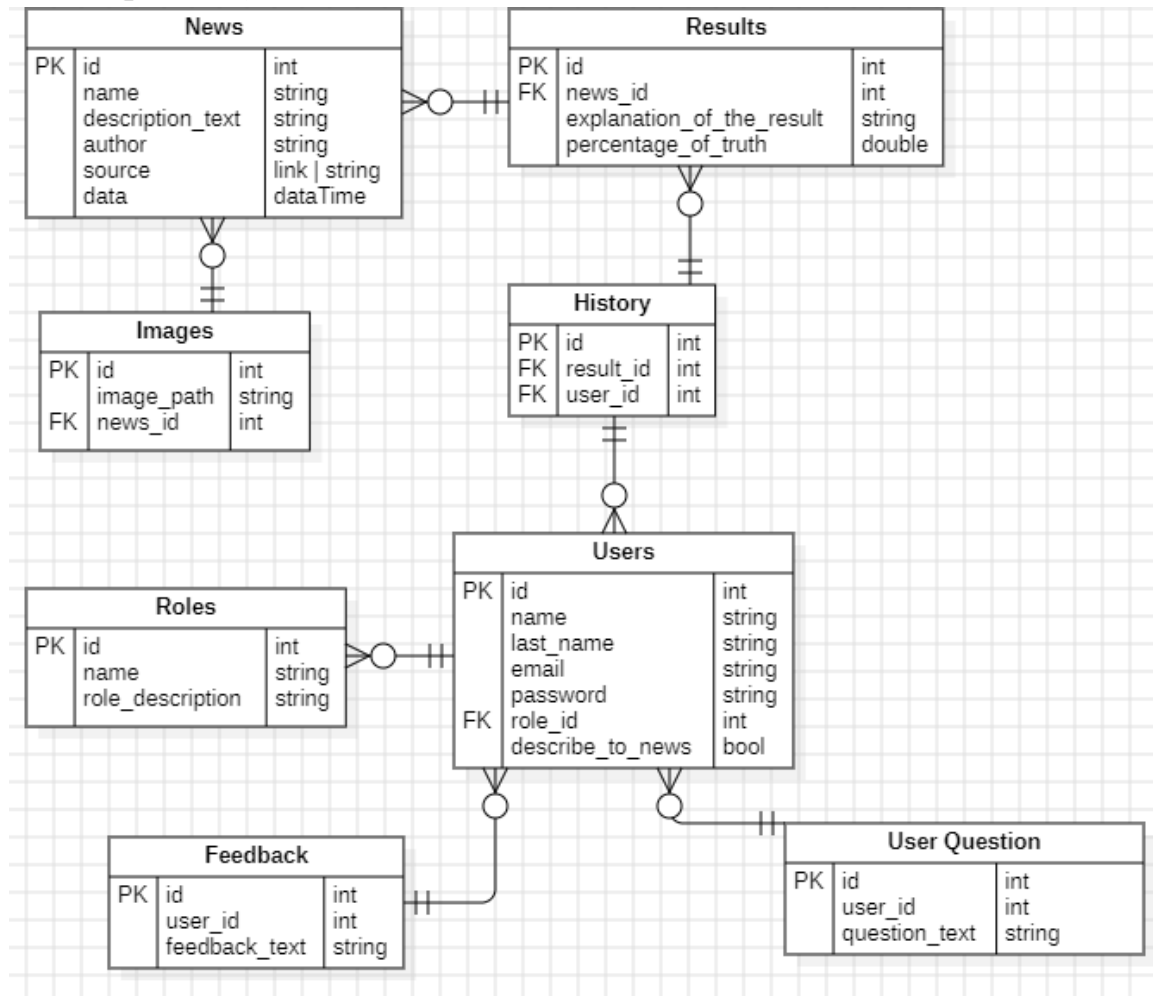


Рис. 2.1 – Початкова UML-діаграма бази даних для веб-додатку виявлення та аналізу фейків

Функціями веб-додатку є:

- ✖ Реєстрація користувачів – дозволяє користувачам створювати облікові записи шляхом надання необхідних даних (ім'я, електронна пошта, пароль) і зберігання їх у таблиці "users".
- ✖ Автентифікація та авторизація – після реєстрації користувачі отримують токен для автентифікації. Пароль користувачів також хешується для безпеки. Це дозволяє авторизованим користувачам використовувати сайт.
- ✖ Перегляд актуальних новин – після авторизації на сайті на головному екрані користувач може побачити актуальні новини в світі.
- ✖ Перегляд історії користувача – після того як ми перевірили новину вона заноситься в таблицю під назвою «Results». Далі користувач може переглянути свою історію своїх перевірок для простоти пошуку ми додали таблицю «History» в якій ми зберігаємо «user_id» та «result_id».
- ✖ Перевірка новини — дозволяє користувачеві перевірити новину на те наскільки вона правдива надавши наступні дані про цю новину (назва новини, основний текст новини, (за наявності) автор новини та посилання звідки надана ця новина)

2.1.2 Опис роботи системи

Для того, щоб користуватися веб-додатком, користувачу потрібно зареєструватися в додатку. Для цього потрібно розробити запит на реєстрацію користувача. Процес реєстрації (рис. 2.2) відбувається таким чином:

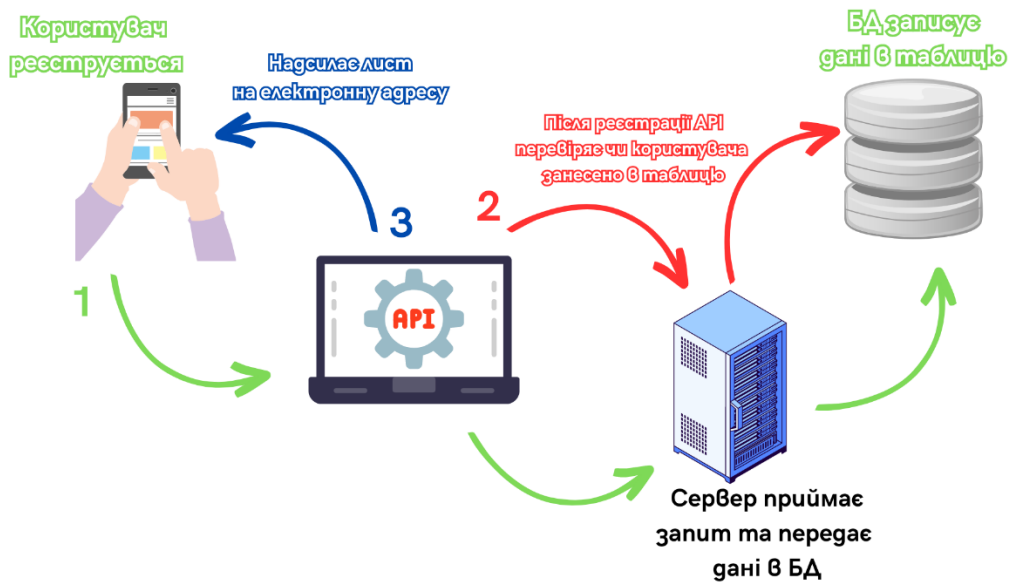


Рис. 2.2 – Процес реєстрації нового користувача

Коли користувач реєструється він надсилає запит на реєстрацію з своїми даними на API вона в свою чергу перевіряє ці данні та надсилає результат який отримує користувач. Приклад заповнення даних реєстрації(рис. 2.3) – “Ім`я”, “пошту” та “пароль”

Sign Up

Yura

Trytenko

yuratrytenko@gmail.com

.....

.....

Sign Up

OR

Already have an account? [Log in](#)

Рис. 2.3 – Приклад заповнення даних реєстрації

API повинно перевірити правильність цих даних, наприклад, чи відповідає пароль мінімальним вимогам – мати не менше ніж 8 символів, чи введена електронна адреса є дійсно електронною адресою. Після обробки, якщо не виникає помилок, дані користувача записуються у таблицю “users” (рис. 2.4) у відповідні поля.

3	1003	Yura	Trytenko	yuratrytenko@gmail.com	Yura12345	NULL	2024-11-25 21:28:59.157
---	------	------	----------	------------------------	-----------	------	-------------------------

Рис. 2.4 – Приклад відображення користувача в таблиці

На рисунку ми бачимо, що нашого користувача успішно було зареєстровано.

Після успішної реєстрації, користувачу потрібно авторизуватися. Для цього використовується сторінка авторизації. Користувач вводить свої дані та при натисканні на кнопку сервер надсилає дані на сервер. Сервер перевіряє, чи існує користувач з вказаною електронною поштою та паролем. Якщо такого користувача не знайдено, повертається помилка.

Коли користувача було авторизовано, він зможе переглянути актуальні новини на головній сторінці сайту. Далі користувач може перейти до перевірки новини на правдивість. Якщо користувач заповнив всі поля правильно то система видасть йому результат в відсотковому співвідношенні та опис чого модель надала саме такий результат і після успішного завершення аналізу результат буде занесено в таблицю «Results». Далі користувач може переглянути історію всіх своїх перевірок які він робив.

2.2 Аналіз та вибір технологій для веб-додатку

Проводячи аналіз сучасних засобів розробки та технологій для веб-додатку виявлення та аналізу фейків в інтернеті, я вибрав наступні компоненти і технології: Visual Studio, GitHub, Session, C#, Asp.Net, Ml.Net, SqlClient.

Середовище програмування Visual Studio є популярним середовищем для розробки веб-додатку з наступними перевагами:

- ✱ підтримує мови C#, C++, Python, JavaScript, TypeScript, F#, Visual Basic та інші, що робить його універсальним середовищем для розробників з різними знаннями. Це полегшує розробку, обмін кодом та співпрацю між розробниками.
- ✱ Завдяки активній спільноті розробників доступна велика кількість плагінів, розширень і шаблонів, які можна використовувати для швидкої інтеграції нових функцій.

- ✱ Visual Studio має вбудований відладчик, профайлер і інструменти для юніт-тестування, що полегшує створення якісного програмного забезпечення.
- ✱ Підтримка Git і інших систем дозволяє легко працювати в команді та синхронізувати код між розробниками.

GitHub — це веб-платформа, що надає можливість хостингу репозиторіїв Git. Вона дозволяє зберігати та управляти версіями коду за допомогою системи контролю версій Git. Завдяки зручному веб-інтерфейсу, GitHub сприяє ефективній співпраці розробників, пропонуючи інструменти для управління проєктами, колаборації, огляду та затвердження змін, а також вирішення проблем і відстеження задач. Крім того, платформа підтримує проєкти з відкритим кодом, забезпечуючи доступ до вихідного коду і залучення нових учасників. Сьогодні GitHub є однією з найпопулярніших платформ для спільної розробки програмного забезпечення.

Session - Сесії є потужним механізмом для збереження інформації про стан користувача протягом його взаємодії із веб-додатком.

- ✱ Збереження стану: Дозволяє зберігати дані між запитами, що корисно для автентифікації, збереження кошика покупок або інших даних користувача.
- ✱ Безпека: Дані зберігаються на сервері, що робить їх менш вразливими до викрадення, ніж куки.
- ✱ Гнучкість: Підтримує різні методи зберігання (пам'ять, бази даних, файлові системи).
- ✱ Простота у використанні: Інтегрується з платформами, такими як ASP.NET, з мінімальними налаштуваннями.

C# — це об'єктно-орієнтована мова програмування від Microsoft, яка використовується для створення додатків різного типу: веб, десктопних, мобільних і хмарних. Завдяки підтримці .NET і сучасному синтаксису є популярним вибором для розробників корпоративних рішень.

ASP.NET — це фреймворк для розробки веб-додатків і API. Підтримує швидку обробку запитів, вбудовану безпеку та розподіл функціоналу за принципом MVC (Model-View-Controller). Інтегрується з хмарними сервісами та іншими технологіями Microsoft.

ML.NET — це фреймворк для машинного навчання, який дозволяє створювати та впроваджувати моделі в .NET-додатки. Підходить для вирішення завдань класифікації, регресії, рекомендацій, а також для обробки тексту і даних. Має підтримку AutoML для автоматизації навчання моделей.

SqlClient — це бібліотека для роботи з базами даних SQL Server у середовищі .NET. Вона дозволяє виконувати запити, виклики збережених процедур і управляти транзакціями з високою швидкістю та надійністю. Ключові переваги перед іншими бібліотеками.

- Простота: Інтуїтивний API дозволяє легко виконувати операції з базою даних.
- Безпека: Підтримує захищене з'єднання через SSL/TLS, а також параметризовані запити для захисту від SQL-ін'єкцій.
- Продуктивність: Оптимізований для взаємодії з SQL Server, що забезпечує швидке виконання запитів.
- Гнучкість: Підходить для різних масштабів, від невеликих проектів до великих корпоративних додатків.
- Інтеграція: Працює в парі з ORM, такими як Entity Framework, а також може використовуватися безпосередньо для точного налаштування запитів.

Висновки до розділу 2

У результаті аналізу та вибору технологій для розробки веб-додатку, призначеного для виявлення та аналізу фейків в інтернеті, був обраний стек технологій, який найкраще відповідає поставленим завданням.

У цьому розділі детально описано функціональні можливості системи, її структуру та принципи роботи. Окрему увагу приділено опису процесу передачі даних у базу даних, їх збереження та подальшого відображення для користувача. Також наведено приклад практичної реалізації цих процесів, що ілюструє узгоджену роботу компонентів системи.

РОЗДІЛ 3.

ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Засоби розробки

Розробку веб-додатку для виявлення та аналізу фейків в інтернеті виконано на мові програмування C# та середовищі програмування Visual Studio.

Платформою для зберігання та керування проектами, з використанням системи контролю версій Git обрано GitHub.

У програмний код було імпортовано такі бібліотеки:

1. Session — це механізм для збереження даних користувача між запитами у веб-додатках, розроблених на основі ASP.NET. Використовується для тимчасового збереження інформації, такої як статус автентифікації або налаштування користувача.
2. Routing — це компонент ASP.NET Core, який дозволяє обробляти URL-запити та спрямовувати їх до відповідних методів контролера чи обробників. Забезпечує гнучкість у налаштуванні маршрутів і підтримує параметризовані маршрути.
3. SqlClient — бібліотека .NET для взаємодії з базами даних SQL Server. Дозволяє виконувати запити, виклики збережених процедур і керувати транзакціями. Підтримує параметризовані запити для безпечної роботи з даними.
4. UseAuthorization — це компонент у середовищі ASP.NET Core, який забезпечує виконання авторизації для запитів. Його використання дозволяє захистити ресурси додатка та перевіряти доступ відповідно до політик авторизації.
5. Базові бібліотеки .NET — це набір бібліотек, які надають основні функції для розробки додатків на C#.

До них належать:

System.Collections — для роботи з колекціями даних.

System.IO — для роботи з файлами та потоками.

System.Net.Http — для роботи з HTTP-запитами.

System.Linq — для спрощення обробки даних через запити.

System.Threading — для роботи з потоками та асинхронними операціями.

3.2 Опис файлової структури проєкту

Шукаючи яку структуру проєкту краще використовувати я зрозумів що краще всього використовувати патерн MVC (Model-View-Controller). Даний патерн забезпечує структуру з чітким розділенням які дані обробляє кожна з частин:

- ✖ Model – відповідає за дані, управління та їх логіку
- ✖ View – відображає всю потрібну інформацію для користувача
- ✖ Controller – приймає запит від користувача, керує моделями і передає відповідні дані в View

Завдяки такий структурі код легше розуміти, підтримувати та тестувати. Також чому мною був обраний саме цей патерн майбутнім розробникам набагато легше буде розібратися в проєкті в якому все розбито на окремі частини.

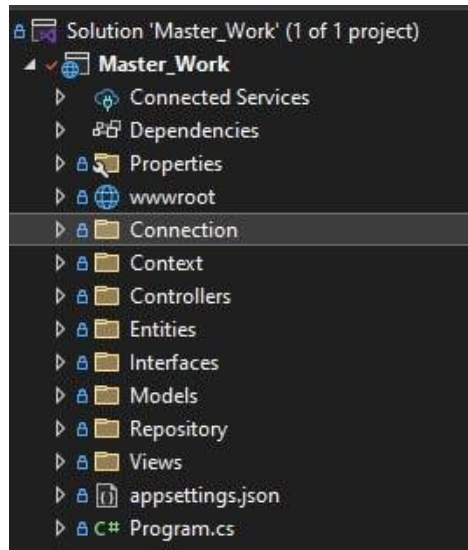


Рис. 3.1 Структура проєкту використовуючи патерн MVC

Структура проєкту містить наступні елементи:

- ✖ Dependencies – містить всі підключаємі в проєкт бібліотеки, пакети та фреймворки які можуть знадобитися розробнику під час створення продукту.
- ✖ Директорія Properties – містить налаштування запуску додатку.
- ✖ Директорія wwwroot – містить такі елементи як файли мови розмітки документів css та файли js(JavaScript) які використовуються для створення сценаріїв веб сторінки.
- ✖ Директорія Connection – містить файли за допомогою яких відбувається звернення та підключення до бази даних.
- ✖ Директорія Controller – директорія містить всі контролери які використовуються в проєкті. З їх допомогою ми зв'язуємо моделі з представленням
- ✖ Директорія Entities – директорія в якій знаходяться всі моделі даних які є в програмі.

- ✖ Директорія Interfaces – дана директорія містить всі інтерфейси нашого проєкту.
- ✖ Директорія Models – директорія в якій зберігається наша модель для аналізу та виявленню фейків.
- ✖ Директорія Repository – містить файли які описують запити та параметри потрібні для успішного звернення в базу даних
- ✖ Директорія Views – директорія в якій зберігаються всі представлення нашого проєкту.
- ✖ Appsettings.json – файл в якому зберігаються всі налаштування нашого проєкту такі як налаштування логів, налаштування рядка підключення в базу даних і т.д.
- ✖ Program.cs – файл в якому налаштовуються DI контейнери

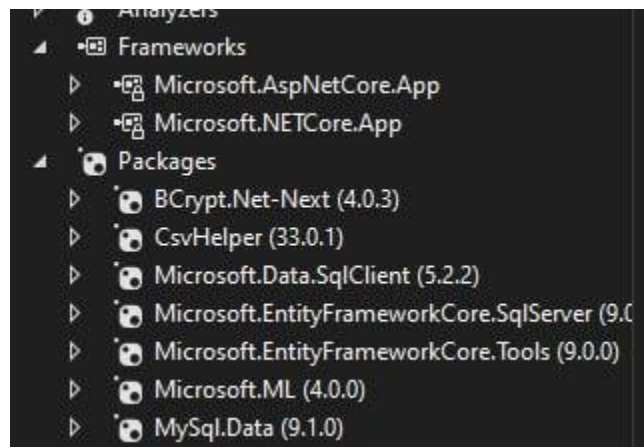


Рис. 3.2 Фреймворки та бібліотеки підключені в проєкті

В проєкті підключені наступні фреймворки:

- ✖ Microsoft.AspNetCore.App та Microsoft.NETCore.App – ці фреймворки надають змогу використовувати базовий функціонал веб технологій Asp.Net на платформі .NetCore.

В проєкті є підключені наступні бібліотеки:

- ✖ Bcrypt – бібліотека яка надає легкий спосіб хешування даних.
- ✖ Csv – допомагає в роботі з файлами типу csv, json і т.д.
- ✖ Microsoft.Data.SqlClient – надає можливість легкий та зручний спосіб працювати з SQL Server
- ✖ Microsoft.Ml – бібліотека яка використовується для створення та впровадження моделі в проєкт.

3.3 Опис реалізації системи

Дизайн сайту було розроблено в Figma. Переглянувши сайти багатьох компаній стало зрозуміло що зараз більшість сайтів розробляють мінімалістичний та інтуїтивно зрозумілий інтерфейс для легкого розуміння куди і як саме потрібно натискати. В своїй роботі я старався зробити інтерфейс максимально мінімалістичним та інтуїтивно зрозумілим як для людини яка не розбирається в компютерах так і для користувачів в яких вже є великий досвід користування технікою за плечима. Отож перейдемо до розгляду кожної з сторінок нашого сайту:

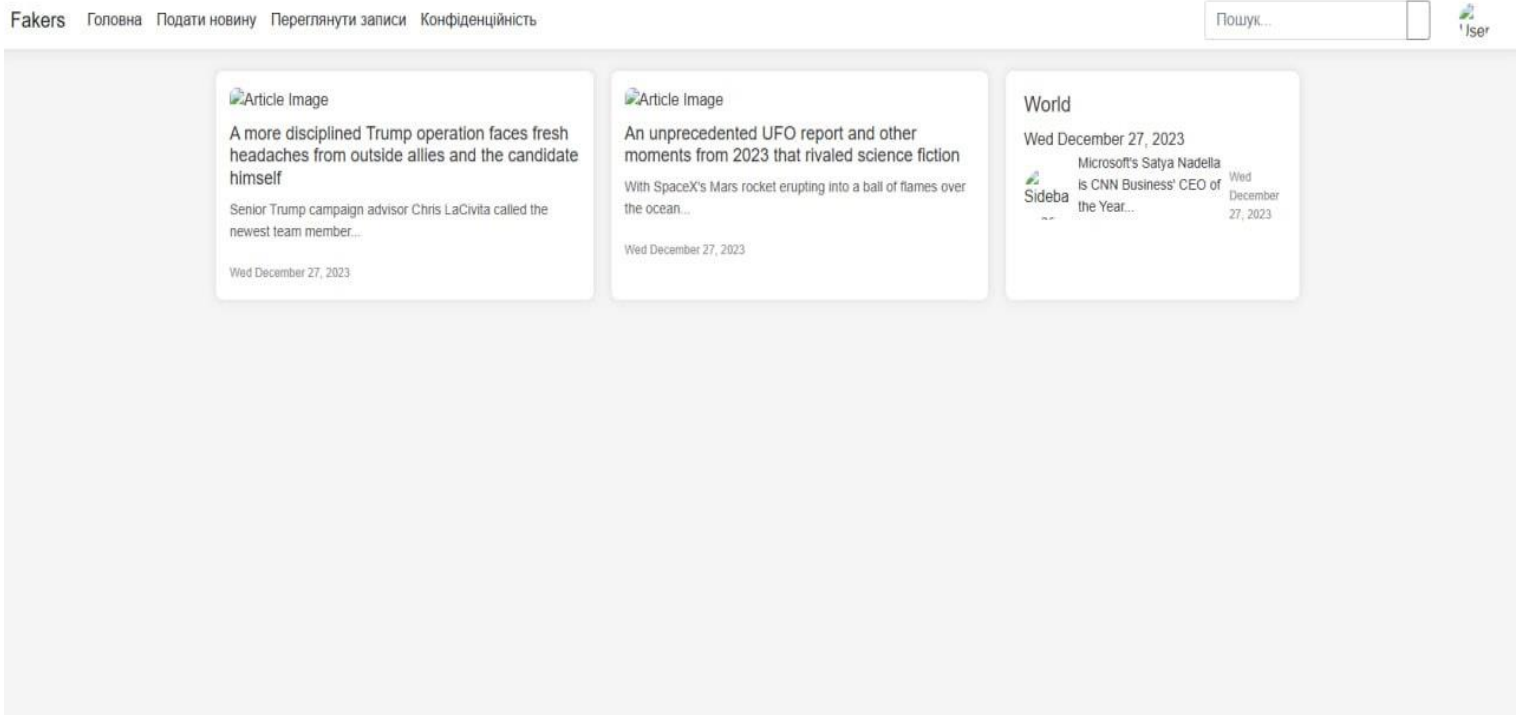


Рис. 3.3 – Дизайн головної сторінки сайту

На рисунку 3.3 ми бачимо головну сторінку з якої і починає знайомство з нашим сайтом користувач.

На даній сторінці ми бачимо топ-бар на якому знаходяться всі самі головні кнопки такі як: Назва сайту, кнопка повернення на головну сторінку, кнопка переходу на сторінку перевірки новини на фейковість, кнопка переходу на сторінку перегляд історії перевірених новин, перехід на сторінку конфіденційності сайту, пошук по новинам та іконку входу на наш сайт.

Далі розглянемо сторінку завдяки якій користувач зможе увійти в свій обліковий запис(рис 3.4)

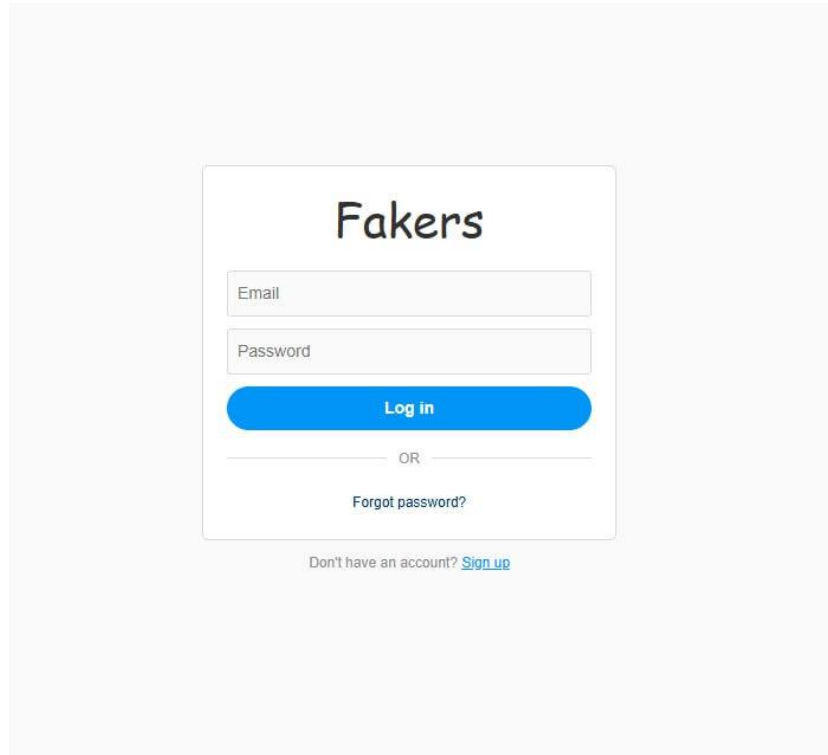
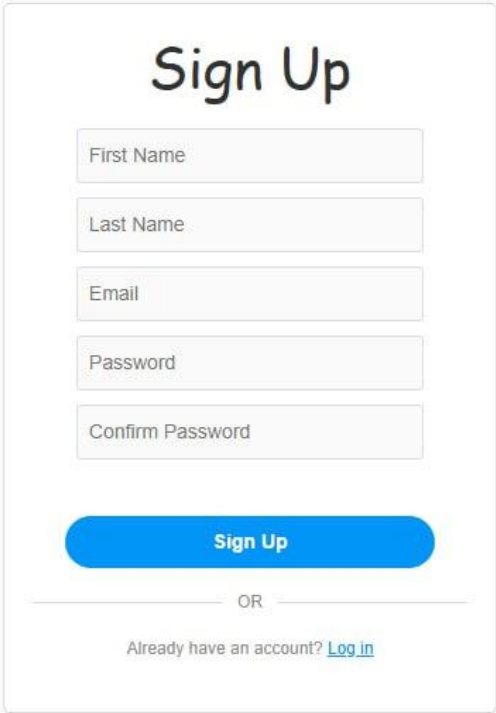


Рис. 3.4 – Дизайн сторінки входу в аккаунт користувача

На даній сторінці ми бачимо сторінку входу на якій користувач має ввести свої данні та увійти в аккаунт. Якщо у користувача немає аккаунту він може натиснути на кнопку SignUp та перейти на сторінку реєстрації та створити аккаунт. Після натискання на кнопку SignUp нас переадресує на сторінку реєстрації в нашій системі і користувач побачить сторінку зображену на рисунку 3.5

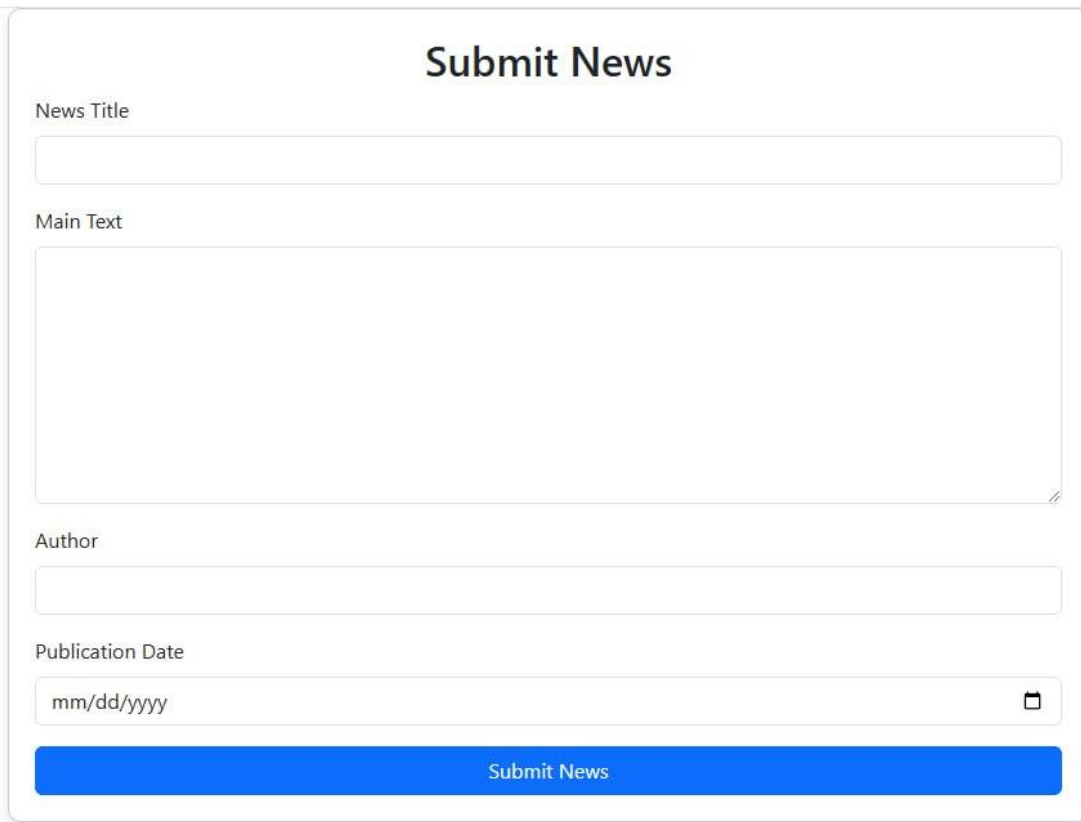


The image shows a 'Sign Up' form centered on a light gray background. The form is enclosed in a white box with a thin gray border. At the top of the box, the text 'Sign Up' is displayed in a large, bold, black font. Below this, there are five input fields stacked vertically, each with a light gray border and a placeholder text: 'First Name', 'Last Name', 'Email', 'Password', and 'Confirm Password'. Below the input fields is a prominent blue button with rounded corners and the text 'Sign Up' in white. Underneath the button, the word 'OR' is centered, flanked by horizontal lines. At the bottom of the form, the text 'Already have an account?' is followed by a blue, underlined link that says 'Log in'.

Рис. 3.5 – Дизайн сторінки реєстрації

На сторінці ми бачимо сторінку на якій користувач може зареєструвати собі аккаунт заповнивши відповідні поля.

Далі перейдемо до сторінки перевірки новини на правдивість. Сторінка перевірки новини зображена на рисунку 3.6:



The image shows a web form titled "Submit News". It contains four input fields: "News Title" (a single-line text box), "Main Text" (a large multi-line text area), "Author" (a single-line text box), and "Publication Date" (a date picker showing "mm/dd/yyyy"). At the bottom of the form is a prominent blue button labeled "Submit News".

Рис. 3.6 – Дизайн сторінки перевірки новини на правдивість

На даному рисунку ми можемо побачити сторінку для перевірки новини. На цій сторінці користувач бачить всю інформацію яку потрібно надати для того щоб отримати відповідь чи дана новина правдива чи ні. Після перевірки користувач може переглянути історію своїх перевірок. Сторінка історії перевірок користувача зображена на рисунку

3.7:

Список записів				
Id	Name	Description	Is Fake	Probability
1	House Intelligence Committee Chairman Devin Nunes is going to have a bad day	He's been under the assumption, like many of us, that the Christopher Steele-dossier was what prompted the Russia investigation so he's been lashing out at the Department of Justice and the FBI in order to protect Trump.	true	0.9975057
2	President Donald Trump said	President Donald Trump said he believed the financial markets have not fully absorbed the tax cut changes in legislation expected to be passed by Congress on Wednesday.	false	0.0036868753

Рис. 3.7 Зображення сторінки перегляду історії перевірок користувача

На даному рисунку ми бачимо всі новини які перевіряв наш користувач. Також ми можемо побачити весь результат який надала нам наша модель.

Далі розглянемо кожну сторінку більш детально. Але основну увагу я би хотів приділити сторінці яка зображена на рисунку 3.3 тобто на сторінці головного екрану користувача на якій користувач буде проводити основну діяльність.

На даній сторінці користувач буде проводити основну роботу в нашій програмі це:

- ✖ Перегляд актуальних новин
- ✖ Перевірка новин на правдивість
- ✖ Вхід та реєстрація в аккаунт
- ✖ Перегляд історії перевірок

Користувач одразу як перейде на наш сайт зможе ознайомитися з самими актуальними новинами. Далі користувачу потрібно увійти в свій аккаунт. Це не обов'язково через те що і не зареєстрований користувач може користуватися функцією перевірки новини але такий користувач не зможе переглянути історію своїх перевірок.

На сторінках входу(рис. 3.4) та реєстрації(рис. 3.5)користувачу потрібно ввести якийсь набір даних в залежності чи він новий користувач чи вже зареєстрований. Якщо користувач новий він повинен надати наступні дані для реєстрації (Ім'я, Фамілію, Пошту та Пароль) якщо користувача вже зареєстровано то від нього потребується лише дійсні (Пошту та Пароль).

На рисунку 3.6 користувачу потрібно заповнити наступні поля (Назва новини — наприклад «Чемпіон світу 2024 Макс Ферстапен», основний текст новини — наприклад «Макс Ферстапен 4 раз підряд чемпіоном світу в формулі 1», автор новини — наприклад «Новини автоспорту України» та дата публікації цієї новини «24.11.2024»). Після заповнення всіх полів користувачу потрібно натиснути на кнопку «Submit News» після чого внизу кнопки з'явиться вікно з результатом перевірки цієї новини яку надала наша модель.

На останньому рисунку 3.7 зображено історію користувача який вже зробив два запити на перевірку новин. На даній сторінці ми бачимо всю потрібну інформацію про перевірену новину (назва новини, основний текст новини, автор цієї новини, чи є дана новина правдива чи ні та процент правдивості новини який нам надала модель)

3.4 Аналіз отриманих результаті

Розглянемо детальніше основні процеси роботи з сайтом такі як реєстрація, авторизація, перевірка новини на правдивість, саму модель перевірки новин на фейковість, роботу з базою даних та перегляд історії користувача .

Почнемо з реєстрації користувач вводить дані в всі доступні поля такі як(Ім'я, Прізвище, Пошту та пароль який він обов'язково повинен підтвердити ввівши 2 раз щоб точно бути впевненим що не допустився помилки вводячи пароль). Сервер надсилає запит в базу даних на додавання нового користувача в базу даних та чекає на відповідь. Якщо і базі даних не виникло ніяких проблем то користувача додасть до таблиці та повідомить користувача що він успішно створив аккаунт якщо ні то користувача видасть чому база даних не змогла занести його в таблицю і попросить виправити дані або спробувати пізніше.

```

public Task<int> AddUserAsync(string name, string email, string lastName, string password)
{
    string hashedPassword = BCrypt.Net.BCrypt.HashPassword(password);

    string query = $"INSERT INTO Users (Name, Email, Last_Name, Password) VALUES (@Name, @Email, @Last_Name, @Password)";
    var parameters = new Dictionary<string, object>
    {
        { "@Name", name },
        { "@Email", email },
        { "@Last_name", lastName },
        { "@Password", hashedPassword }
    };

    return _cFactory.ExecuteNonQueryAsync(query, parameters);
}

```

Рис. 3.8 Створення запиту на реєстрацію з даними які надав користувач

```

public Task<int> ExecuteNonQueryAsync(string query, Dictionary<string, object>? parameters = null)
{
    using (var connection = new SqlConnection(_connectionString))
    {
        connection.Open();

        using (var command = new SqlCommand(query, connection))
        {
            if (parameters != null)
            {
                foreach (var param in parameters)
                {
                    command.Parameters.AddWithValue(param.Key, param.Value);
                }
            }

            return command.ExecuteNonQuery();
        }
    }
}

```

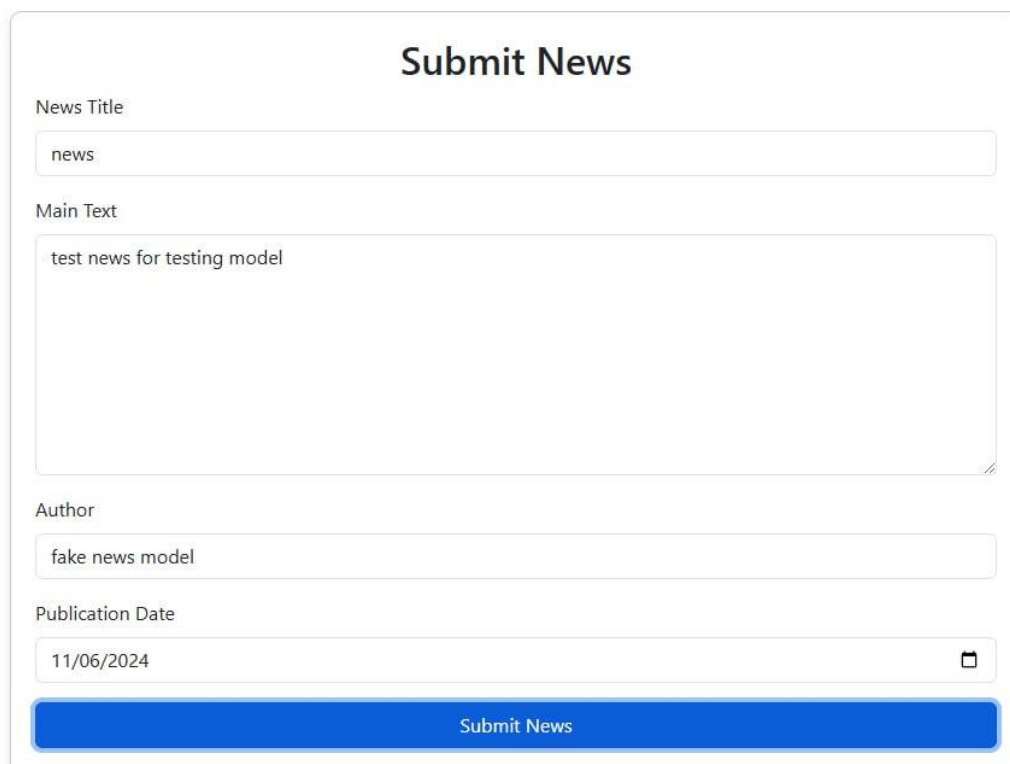
Рис. 3.9 Надсилення запиту на реєстрацію

На рисунку 3.8 ми можемо бачити що наш метод який ми використовуємо для створення контексту того що ми будемо передавати в нашу базу даних. В цьому методі ми приймає в якості свої параметрів всі дані які потрібні користувачу для реєстрації. Далі для більшої захищеності нашої системи ми хешуємо пароль користувача. Наступним кроком ми створюємо запит в нашу базу даних за допомогою Sql запиту в який ми передаємо всі наші вхідні параметри які нам надійшли від користувача. Кінцевим кроком цього методу є виклик методу який додасть наші дані в базу даних зображений на рисунку 3.9. На даному рисунку зображено підключення до самої бази даних та внесення даних в неї. Результат роботи цих двох методів продемонстровано на рисунку 3.10.

1007	Yura	Trytenko	yuraTrytenko@gmail.com	\$2a\$11\$3uD/CAtG4GuvcbnxsHsK/e1qS2wbtODH7OKcV0yko...	NULL
------	------	----------	------------------------	--	------

Рис. 3.10 Створений користувач в базі даних

Після успішної реєстрації користувач може увійти в свій обліковий запис. Далі перейдемо до основного етапу нашої програми це перевірка новин на правдивість. Приклад заповнення даних продемонстровано на рисунку 3.11



The image shows a web form titled "Submit News". It contains four input fields and a submit button. The first field, "News Title", contains the text "news". The second field, "Main Text", is a larger text area containing "test news for testing model". The third field, "Author", contains "fake news model". The fourth field, "Publication Date", contains "11/06/2024" and has a calendar icon on the right. At the bottom is a blue button labeled "Submit News".

Рис. 3.11 Приклад заповнення даних всіх полів на сторінці перевірки новини на правдивість

Після успішного заповнення всіх полів користувачу потрібно натиснути кнопку Submit News. Після натискання кнопки користувач побачить віконечко яке з'явилося нижче кнопки там де будуть результати перевірки новини на правдивість. Результати перевірки зображені на рисунку 3.12

Перевірка новини

Заголовок новини

news

Основний текст

test news for testing model

Автор

prestige

Дата публікації

12/24/2024

Надіслати новину

Результати прогнозування:

Чи є фейковою: true

Імовірність: 0.9961665

Правдивість: 0.38%

Рис. 3.12 Результат виконання аналізу новини та відображення результату користувачу.

На даному рисунку ми бачимо що новина є правдивою та що модель гадає що ця новина правдива на 67.3%.

Почнемо з огляду моделі як вона працює та які методи реалізовано. Почнемо з огляду моделей які нам потрібні для реалізації нашої моделі. Дані моделі зображені на рисунках 3.13, 3.14, 3,15.

```

public class NewsData
{
    [LoadColumn(0)]
    public string Title { get; set; }

    [LoadColumn(1)]
    public string Content { get; set; }

    [LoadColumn(2)]
    public string Author { get; set; }

    [LoadColumn(3)]
    public string PublicationDate { get; set; }
}

public class NewsPrediction
{
    [ColumnName("PredictedLabel")]
    public bool IsFake { get; set; }
    public float Probability { get; set; }
    public float Score { get; set; }
}

```

Рис. 3.13 Реалізація моделей NewsData та NewsPrediction

Це дві основні моделі які приймають участь у тренуванні та прогнозуванні даних.

Метод NewsData – це метод який використовується для того щоб отримати дані з файлів True.csv та Fake.csv.

Метод NewsPrediction – в моделі він використовується як метод який визначає яку відповідь повинен отримати наш користувач. З цього методу виходить що наш користувач отримає наступну відповідь від нашої моделі: IsFake = false, Probability = 0,9614234, Score = 0,8923423.

```

public class NewsArticle
{
    public string title { get; set; }
    public string text { get; set; }
    public string subject { get; set; }
    public string date { get; set; }
}

```

Рис. 3.14 Реалізація моделі NewsArticle

Модель NewsArticle використовується вже коли в нас є навчена модель і ми хочемо перевірити роботу нашої моделі. В ці змінні ми вже

зберігаємо дані надані нам користувачем та використовуємо цю модель виключно для аналізу.

```
public class TrainingData
{
    public string Title { get; set; }
    public string Content { get; set; }
    public string Author { get; set; }
    public string PublicationDate { get; set; }
    public bool Label { get; set; } // True для фейкових, False для правдивих
}
```

Рис. 3.15 Реалізація моделі TrainingData

Дана модель використовується безпосередньо в тренуванні моделі. Коли ми зчитали данні і збереги їх за допомогою моделі NewsData нам потрібно додати до них пометку які новини є фейкові а які є правдиві. Саме для цієї мети ми створили дану модель.

Під час створення моделей ми стикнулися з наступною проблемою в деяких стовпчиках дані могли бути хибні наприклад в рядку з датою ми могли мати посилання або текст тому було прийняте рішення створити додатковий метод для вирішення таких проблем.

```
public void SaveModel(string modelPath)
{
    _mlContext.Model.Save(_model, null, modelPath);
}

public void LoadModel(string modelPath)
{
    _model = _mlContext.Model.Load(modelPath, out _);
}

// Remove links using a regular expression
private string RemoveLinks(string text)
{
    if (string.IsNullOrEmpty(text)) return text;

    // Regular expression to match URLs
    var linkPattern = @"https?:\/\/[^\s]+";
    return Regex.Replace(text, linkPattern, string.Empty);
}
```

Рис. 3.16 – Методи для вирішення проблем з конвертацією полів

Метод RemoveLinks метод який допомагає уникнути ситуації коли в нас в стовпчику з датою можуть потрапити посилання. Також на рисунку 3.16 є два методи SaveModel і LoadModel вони також відіграють дуже важливу роль. Через те що навчаємо та

використовуємо модель в різний час постало питання про збереження навченої моделі та використання її в будь який час. Метод `SaveModel` дозволяє зберегти нам модель в архів і використовувати її будь який зручний для нас час. `LoadModel` в свою чергу дозволяє завантажити навчену модель в будь який час та використовувати її.

```
// Check and parse date
private string ParseDate(object date)
{
    if (date is string dateString)
    {
        if (DateTime.TryParse(dateString, out DateTime parsedDate))
        {
            return parsedDate.ToString("yyyy-MM-dd");
        }
        else
        {
            return string.Empty;
        }
    }
    else
    {
        return string.Empty;
    }
}
```

Рис. 3.17 Допоміжний метод `ParseDate`

`ParseDate` також в нас допоміжний і він допомагає тим що якщо в нас наприклад пустий рядок або значення `bool` в рядку цей метод допомагає уникнути помилки. І в випадку якщо там дані які нам не вдається конвертувати в дату то ми робимо це поле пустим і тим самим уникаємо ще декількох помилок які могли уникнути через не валідні дані.

```

public void TrainModel(string trueNewsFilePath, string fakeNewsFilePath)
{
    // Load true and fake news
    var trueNews = NewsFileProcessor.LoadNewsFromFile(trueNewsFilePath)
        .Select(news => new TrainingData
        {
            Title = news.title,
            Content = RemoveLinks(news.text), // Remove hyperlinks from content
            Author = news.subject,
            PublicationDate = ParseDate(news.date),
            Label = false // True news
        });

    var fakeNews = NewsFileProcessor.LoadNewsFromFile(fakeNewsFilePath)
        .Select(news => new TrainingData
        {
            Title = news.title,
            Content = RemoveLinks(news.text), // Remove hyperlinks from content
            Author = news.subject,
            PublicationDate = ParseDate(news.date),
            Label = true // Fake news
        });

    // Merge datasets
    var trainingData = trueNews.Concat(fakeNews).ToList();

    // Convert data to DataView
    var dataView = _mlContext.Data.LoadFromEnumerable(trainingData);

    // Build the pipeline
    var pipeline = _mlContext.Transforms.Text.FeaturizeText("Features", nameof(TrainingData.Content))
        .Append(_mlContext.BinaryClassification.Trainers.SdcaLogisticRegression(labelColumnName: nameof(TrainingData.Label), featureColumnName: "Features"));

    // Train the model
    _model = pipeline.Fit(dataView);
}

```

Рис.3.18 Тренування моделі

В даному методі ми тренуємо модель для аналізу фейкових новин за допомогою ML.Net. Спочатку завантажуються два набори даних: один містить правдиві новини, інший — фейкові. Дані обробляються шляхом видалення гіперпосилань із тексту (метод `RemoveLinks`) та парсингу дат публікації (метод `ParseDate`), а також перетворюються у формат `TrainingData`, який включає заголовок, текст, автора, дату публікації та мітку (`Label`), що позначає тип новини (`false` для правдивих і `true` для фейкових). Обидва набори об'єднуються в один загальний список, який конвертується у формат `IDataView`, необхідний для роботи з ML.Net.

Далі створюється пайплайн, що складається з двох ключових етапів. Перший етап — це трансформація текстових даних у числові ознаки за допомогою функції `FeaturizeText`, яка генерує вектор ознак (`Features`) з тексту новини. Другий етап — це застосування алгоритму логістичної регресії `SdcaLogisticRegression`, який призначений для задач бінарної класифікації. У цьому алгоритмі використовується вхідна колонка `Features` для аналізу тексту і колонка `Label` для позначення типу

новини. На завершальному етапі модель тренується на основі цього пайплайна за допомогою методу `Fit`, що дозволяє отримати готову модель для класифікації новин.

```
public NewsPrediction Predict(NewsArticle article)
{
    if (_model == null)
    {
        throw new InvalidOperationException("Model is not loaded. Train or load the model before making predictions.");
    }

    var predictionEngine = _mlContext.Model.CreatePredictionEngine<TrainingData, NewsPrediction>(_model);

    var data = new TrainingData
    {
        Title = article.title,
        Content = RemoveLinks(article.text),
        Author = article.subject,
        PublicationDate = article.date.ToString()
    };

    var prediction = predictionEngine.Predict(data);

    // Повертаємо результат з ймовірністю
    return prediction;
}
```

Рис. 3.19 Метод використання моделі для передбачення

Даний метод ми використовуємо для передбачення нашої моделі. На початку ми перевіряємо чи модель в нас присутня. Якщо ні то ми видаємо помилку про те що моделі в не тренована і ми не можемо зробити передбачення. Якщо ж модель в нас присутня то ми надаємо їй данні нашого користувача і далі отримуємо відповідь від моделі чи правдива новина чи ні.

Далі буде продемонстровано файл контроллера в якому передаються данні користувача до моделі.

```
[HttpPost("predict")]
public IActionResult Predict([FromBody] NewsArticle article)
{
    var modelPath = Path.Combine(Path.GetTempPath(), "FakeNewsModel.zip");

    if (!System.IO.File.Exists(modelPath))
    {
        return BadRequest("Модель не знайдена. Спочатку натренуйте модель.");
    }

    _newsAnalyzer.LoadModel(modelPath);
    var prediction = _newsAnalyzer.Predict(article);

    // Повертаємо ймовірність та результат
    return Ok(new
    {
        IsFake = prediction.IsFake,
        Probability = prediction.Probability,
        TruthfulnessPercentage = (1 - prediction.Probability) * 100 // Відсоток правдивості
    });
}
```

Рис. 3.20 Контролер передбачення

В данному контроллері ми передаємо навчену модель в наш метод який зображено на рисунку 3.19. з усіма даними для того щоб наша модель змогла зробити передбачення.

```
[HttpPost("train")]
public IActionResult TrainModel([FromForm] IFormFile trueNewsFile, [FromForm] IFormFile fakeNewsFile)
{
    var trueNewsPath = Path.GetTempFileName();
    var fakeNewsPath = Path.GetTempFileName();
    var modelPath = Path.Combine(Path.GetTempPath(), "fakeNewsModel.zip");

    using (var stream = new FileStream(trueNewsPath, FileMode.Create))
    {
        trueNewsFile.CopyTo(stream);
    }

    using (var stream = new FileStream(fakeNewsPath, FileMode.Create))
    {
        fakeNewsFile.CopyTo(stream);
    }

    _newsAnalyzer.TrainModel(trueNewsPath, fakeNewsPath);
    _newsAnalyzer.SaveModel(modelPath);

    return Ok("Model trained and saved successfully.");
}
```

Рис. 3.21 Контроллер навчання

В даному контроллері ми передаємо два файли (один із правдивими новинами, інший із фейковими) в метод TrainModel, який зображено на рисунку. Метод обробляє ці файли, зберігаючи їх у тимчасових директоріях, і передає їх у процес навчання моделі. Після цього модель зберігається у файл fakeNewsModel.zip, щоб її можна було використовувати для подальших передбачень.

На останок пропоную розглянути декілька прикладів роботи програми. Розпочнемо з декількох прикладів завідомо правдивих новин.

Перевірка новини

Заголовок новини

As U.S. budget fight looms, Republicans flip their fiscal script

Основний текст

their leverage in the Senate, which Republicans narrowly control, to defend both discretionary non-defense programs and social spending, while tackling the issue of the "Dreamers," people brought illegally to the country as children. Trump in September put a March 2018 expiration date on the Deferred Action for Childhood Arrivals, or DACA, program, which protects the young immigrants from deportation and provides them with work permits. The president has said in recent Twitter messages he wants funding for his proposed Mexican border wall and other immigration law changes in exchange for agreeing to help the Dreamers. Representative Debbie Dingell told CBS she did not favor linking that issue to other policy objectives, such as wall funding. "We need to do DACA clean," she said.

Автор

politicsNews

Дата публікації

12/31/2017

Надіслати новину

Результати прогнозування:

Чи є фейковою: false

Імовірність: 0.06287153

Правдивість: 93.71%

Рис. 3.22 Приклад №1 перевірки новини

В цьому прикладі надана правдива новина. Ця новина базується на достовірних фактах, таких як реальні висловлювання представників Конгресу США, включаючи Марка Мідоуза, Джозефа Кроулі та інших, а також конкретні політичні події, зокрема обговорення бюджету, податкової реформи та соціальних програм. Інформація також підтверджена посиланням на офіційні джерела, такі як програма CBS "Face the Nation". Всі згадані події відповідають реальним політичним процесам у США.

Перевірка новини

Заголовок новини

White House, Congress prepare for talks on spending, immigration

Основний текст

WEST PALM BEACH, Fla./WASHINGTON (Reuters) - The White House said on Friday it was set to kick off talks next week with Republican and Democratic congressional leaders on immigration policy, government spending and other issues that need to be wrapped up early in the new year. The expected flurry of legislative activity comes as Republicans and Democrats begin to set the stage for midterm congressional elections in November. President Donald Trump's Republican Party is eager to maintain control of Congress while Democrats look for openings to wrest seats away in the Senate and the House of Representatives. On Wednesday, Trump's budget chief Mick Mulvaney and legislative affairs director Marc Short will meet with Senate Majority Leader Mitch McConnell and House Speaker Paul Ryan - both

Автор

politicsNews

Дата публікації

12/29/2017

Надіслати новину

Результати прогнозування:

Чи є фейковою: false

Імовірність: 0.034691803

Правдивість: 96.53%

Рис. 3.23 Приклад №2 перевірки новини

В цьому прикладі стаття містить правдиву інформацію. Новина базується на офіційних заявах Білого дому та деталях про реальні політичні події, такі як переговори між представниками Республіканської та Демократичної партій щодо імміграційної політики, фінансування уряду та інших законодавчих ініціатив. Новина також містить факти, що підтверджуються публічними заявами ключових політичних діячів, таких як президент Дональд Трамп, Мік Малвейні, Ненсі Пелосі та інші. У ній описуються конкретні законодавчі дедлайни, наприклад, необхідність ухвалення бюджету до 19 січня, а також питання, що стосуються програми DACA та

підвищення боргового ліміту. Інформація відповідає реальним політичним процесам у США та підкріплена посиланнями на достовірні джерела.

І на останок розглянемо завідомо фейкову новину.

Перевірка новини

Заголовок новини

Drunk Bragging Trump Staffer Started Russian Collusion Investigation

Основний текст

that Russian officials were shopping around possible dirt on then-Democratic presidential nominee Hillary Clinton. Exactly how much Mr. Papadopoulos said that night at the Kensington Wine Rooms with the Australian, Alexander Downer, is unclear, the report states. But two months later, when leaked Democratic emails began appearing online, Australian officials passed the information about Mr. Papadopoulos to their American counterparts, according to four current and former American and foreign officials with direct knowledge of the Australians' role. Papadopoulos pleaded guilty to lying to the F.B.I. and is now a cooperating witness with Special Counsel Robert Mueller's team. This isn't a presidency. It's a badly scripted reality TV show. Photo by Win McNamee/Getty Images.

Автор

News

Дата публікації

12/31/2017 📅

Надіслати новину

Результати прогнозування:

Чи є фейковою: true

Імовірність: 0.9914272

Правдивість: 0.86%

Рис. 3.24 Приклад №3 перевірки новини

Ця стаття є недостовірною через те що вона містить маніпулятивну інформацію. У ній використано маніпулятивний стиль подачі інформації, який має на меті дискредитацію певних політичних фігур, таких як Девін Нуньєс, Дональд Трамп та інші. Основна увага приділяється спотворенню фактів і перебільшенню, наприклад,

зменшенню ролі Джорджа Пападопулоса в кампанії Трампа до рівня "кавового хлопчика", хоча це не підтверджується достовірними джерелами. Заяви про його нібито участь у п'яних розмовах у винному барі також подано без належних доказів, що підриває довіру до матеріалу. Інформація базується на чутках і припущеннях, без перевірених фактів або офіційних підтверджень, що робить її ненадійною та недостовірною.

Висновки до розділу 3

У даному розділі були розглянуті та обрані засоби розробки, описані їхні переваги і аргументи щодо доцільності їхнього використання. Була описана реалізація веб-додатку та як з ним взаємодіяти .

Фінальною частиною розділу була демонстрація роботи веб-додатку, де було продемонстровано основні можливості , як система взаємодіє з користувачем та надано декілька прикладів роботи нашої системи.

Робота веб-додатку та інтерфейсу виконується без помилок. Система є досить гнучкою та має великий потенціал для модифікацій та покращень.

ВИСНОВКИ

В ході проектування та розробки веб-додатку для аналізу та виявленню фейків в інтернеті була досягнута мета, та виконані усі поставлені задачі

Завдання дослідження включали аналіз предметної області, проектування веб-додатку, а також вибір інструментів і методів для його реалізації. Аналіз предметної області охоплював вивчення сучасних технологій та рішень у сфері виявлення фейкових новин в Інтернеті, а також визначення основних потреб і вимог користувачів. На основі цього аналізу було проведено проектування веб-додатку та розроблено інтерфейс користувача.

Під час розробки було здобуто досвід і навички, які допоможуть ефективніше працювати з подібними веб-додатками та спростять подальшу розробку існуючої системи.

Розробка цього додатку виявилася надзвичайно актуальною. Аналіз ринку та аналогічних систем показав, що створення власного додатку дозволить заощадити кошти та забезпечить потенціал для подальшого розвитку і вдосконалення.

Даний веб-додаток може конкурувати з іншими, оскільки передбачає розробку та впровадження інструментів, які можуть бути адаптовані для детекції фейкових новин відповідно до потреб конкретного користувача.

Застосування цього додатку дозволить ефективно виявляти фейкові новини, зручно аналізувати інформацію, розробляти власні моделі для оцінки достовірності новин.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. C# Documentation [Електронний ресурс]. – Режим доступу: <https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/overview>
2. Asp.Net Documentation [Електронний ресурс]. – Режим доступу: <https://learn.microsoft.com/en-us/dotnet/api/?view=aspnetcore-9.0>
3. ML.Net Documentation [Електронний ресурс]. – Режим доступу: <https://learn.microsoft.com/en-us/dotnet/machine-learning/>
4. Марк Джей Прайс C# and .Net 7.
Видання сьоме . Видавництво Packt Publishing, 2022.
5. Адам Фрімен Pro ASP.NET Core 6 Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages [Електронний ресурс]. – Режим доступу: <https://dl.ebooksworld.ir/books/Pro.ASP.NET.Core.6.9th.Edition.Adam.Freeman.Apress.9781484279564.EBooksWorld.ir.pdf>
6. Рекс Хартсон The UX Book: Process and Guidelines for Ensuring a Quality User Experience. Перше видання. Видавництво Morgan Kaufmann, 2012.
7. Git Documentation [Електронний ресурс]. – Режим доступу: <https://git-scm.com/docs>
8. Introduction to ASP.NET SESSION [Електронний ресурс]. – Режим доступу: <https://www.c-sharpcorner.com/UploadFile/225740/introduction-of-session-in-Asp-Net/>

Додаток А. Код програми

Файл ConnectionFactory

```
public class ConnectionFactory
{
    private readonly string _connectionString;

    public ConnectionFactory(string connectionString)
    {
        _connectionString = connectionString;
    }

    public async Task<List<Dictionary<string, object>>> ExecuteQueryAsync(string
query, Dictionary<string, object>? parameters = null)
    {
        var results = new List<Dictionary<string, object>>();

        using (var connection = new MySqlConnection(_connectionString))
        {
            await connection.OpenAsync();

            using (var command = new MySqlCommand(query, connection))
            {
                if (parameters != null)
                {
                    foreach (var param in parameters)
                    {
                        command.Parameters.AddWithValue(param.Key, param.Value);
                    }
                }

                using (var reader = await command.ExecuteReaderAsync())
                {
                    while (await reader.ReadAsync())
                    {
                        var row = new Dictionary<string, object>();
                        for (var i = 0; i < reader.FieldCount; i++)
                        {
                            row[reader.GetName(i)] = reader.GetValue(i);
                        }
                    }
                }
            }
        }
    }
}
```

51

```
        results.Add(row);
    }
}
}

return results;
}

public Task<int> ExecuteNonQueryAsync(string query, Dictionary<string, object>?
parameters = null)
{
    using (var connection = new SqlConnection(_connectionString))
    {
        connection.Open();

        using (var command = new SqlCommand(query, connection))
        {
            if (parameters != null)
            {
                foreach (var param in parameters)
                {
                    command.Parameters.AddWithValue(param.Key, param.Value);
                }
            }

            return command.ExecuteNonQuery();
        }
    }
}
}
```

Файл ApplicationDbContext

```
public class ApplicationDbContext : DbContext
{
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) :
base(options) { }

    public DbSet<Record> Records { get; set; }
```

52

```
}  
}
```

Файл DataController

```
public class DataController : Controller  
{  
    private readonly ConnectionFactory _connectionFactory;  
  
    public DataController(IConfiguration configuration)  
    {  
        var connectionString = configuration.GetConnectionString("DefaultConnection");  
        _connectionFactory = new ConnectionFactory(connectionString);  
    }  
  
    public async Task<IActionResult> History()  
    {  
        string query = "SELECT * FROM History";  
        var records = await _connectionFactory.ExecuteQueryAsync(query); // Заміни на  
назву твоєї таблиці  
  
        return View(records);  
    }  
}
```

Файл HomeController

```
public class HomeController : Controller  
{  
    private readonly ILogger<HomeController> _logger;  
    public readonly GenericRepository _gRepository;  
  
    public HomeController(ILogger<HomeController> logger, GenericRepository  
genericRepository)  
    {  
        _logger = logger;  
        _gRepository = genericRepository;  
    }  
  
    public IActionResult Index()  
    {
```

53

```
        return View();
    }

    public IActionResult LogIn()
    {
        return View();
    }

    public IActionResult NewsPredict()
    {
        return View();
    }

    public IActionResult History()
    {
        return View();
    }

    #region Register
    public IActionResult Register()
    {
        return View();
    }

    [HttpPost]
    public IActionResult Register(SQLRegister model)
    {
        if (ModelState.IsValid)
        {
            _gRepository.AddUserAsync(
                model.FirstName,
                model.Email,
                model.LastName,
                model.Password
            );

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

54

```
        return View(model);
    }

    #endregion

    public IActionResult Privacy()
    {
        return View();
    }

    [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore =
true)]
    public IActionResult Error()
    {
        return View(new ErrorViewModel { RequestId = Activity.Current?.Id ??
HttpContext.TraceIdentifier });
    }
}
```

Файл NewsController

```
[ApiController]
[Route("api/[controller]")]
public class NewsController : ControllerBase
{
    private readonly FakeNewsAnalyzer _newsAnalyzer;

    public NewsController()
    {
        _newsAnalyzer = new FakeNewsAnalyzer();
    }

    [HttpPost("train")]
    public IActionResult TrainModel([FromForm] IFormFile trueNewsFile, [FromForm]
IFormFile fakeNewsFile)
    {
        var trueNewsPath = Path.GetTempFileName();
        var fakeNewsPath = Path.GetTempFileName();
        var modelPath = Path.Combine(Path.GetTempPath(), "fakeNewsModel.zip");
```

```

using (var stream = new FileStream(trueNewsPath, FileMode.Create))
{
    trueNewsFile.CopyTo(stream);
}

using (var stream = new FileStream(fakeNewsPath, FileMode.Create))
{
    fakeNewsFile.CopyTo(stream);
}

_newsAnalyzer.TrainModel(trueNewsPath, fakeNewsPath);
_newsAnalyzer.SaveModel(modelPath);

return Ok("Model trained and saved successfully.");
}

[HttpPost("predict")]
public IActionResult Predict([FromBody] NewsArticle article)
{
    var modelPath = Path.Combine(Path.GetTempPath(), "fakeNewsModel.zip");

    if (!System.IO.File.Exists(modelPath))
    {
        return BadRequest("Модель не знайдена. Спочатку натренуйте модель.");
    }

    _newsAnalyzer.LoadModel(modelPath);
    var prediction = _newsAnalyzer.Predict(article);

    return Ok(new
    {
        IsFake = prediction.IsFake,
        Probability = prediction.Probability,
        TruthfulnessPercentage = (1 - prediction.Probability) * 100 // Відсоток
        правдивості
    });
}

```

56

}

Файл NewsArticle

```
public class NewsArticle
{
    public string title { get; set; }
    public string text { get; set; }
    public string subject { get; set; }
    public string date { get; set; }
}
```

Файл NewsData

```
public class NewsData
{
    [LoadColumn(0)]
    public string Title { get; set; }

    [LoadColumn(1)]
    public string Content { get; set; }

    [LoadColumn(2)]
    public string Author { get; set; }

    [LoadColumn(3)]
    public string PublicationDate { get; set; }
}
```

```
public class NewsPrediction
{
    [ColumnName("PredictedLabel")]
    public bool IsFake { get; set; }
    public float Probability { get; set; }
    public float Score { get; set; }
}
```

Файл Record

```
public class Record
{
    public int Id { get; set; }
```

57

```
public string Name { get; set; }
public string Description { get; set; }
public bool IsFake { get; set; }
public double Probability { get; set; }
}
```

Файл SQLLogin

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

Файл SQLUser

```
public class SQLUser
{
    public class SQLRegister : IEntity<long>
    {
        public long Id { get; set; }
        [Required]
        public string FirstName { get; set; }

        [Required]
        public string LastName { get; set; }

        [Required, EmailAddress]
        public string Email { get; set; }

        [Required, MinLength(6)]
        public string Password { get; set; }

        [Required]
        [Compare("Password", ErrorMessage = "Password and confirmation password do not match.")]
        public string ConfirmPassword { get; set; }
    }
}
```

Файл TrainingData

```
public class TrainingData
{
    public string Title { get; set; }
    public string Content { get; set; }
    public string Author { get; set; }
    public string PublicationDate { get; set; }
    public bool Label { get; set; } // True для фейковых, False для правдивых
}
```

Файл IEntity

```
public interface IEntity<T>
{
    T Id { get; set; }
}
```

Файл ErrorViewModel

```
public class ErrorViewModel
{
    public string? RequestId { get; set; }

    public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);
}
```

Файл FakeNewsAnalyzer

```
public class FakeNewsAnalyzer
{
    private readonly MLContext _mlContext;
    private ITransformer _model;

    public FakeNewsAnalyzer()
    {
        _mlContext = new MLContext();
    }

    public void TrainModel(string trueNewsFilePath, string fakeNewsFilePath)
    {
        // Load true and fake news
        var trueNews = NewsFileProcessor.LoadNewsFromFile(trueNewsFilePath)
            .Select(news => new TrainingData
```

```

    {
        Title = news.title,
        Content = RemoveLinks(news.text),
        Author = news.subject,
        PublicationDate = ParseDate(news.date),
        Label = false
    });

var fakeNews = NewsFileProcessor.LoadNewsFromFile(fakeNewsFilePath)
    .Select(news => new TrainingData
    {
        Title = news.title,
        Content = RemoveLinks(news.text),
        Author = news.subject,
        PublicationDate = ParseDate(news.date),
        Label = true
    });

var trainingData = trueNews.Concat(fakeNews).ToList();

var dataView = _mlContext.Data.LoadFromEnumerable(trainingData);

var pipeline = _mlContext.Transforms.Text.FeaturizeText("Features",
    nameof(TrainingData.Content))
    .Append(_mlContext.BinaryClassification.Trainers.SdcaLogisticRegression(label
    ColumnName: nameof(TrainingData.Label), featureColumnName: "Features"));

_model = pipeline.Fit(dataView);
}

public void SaveModel(string modelPath)
{
    _mlContext.Model.Save(_model, null, modelPath);
}

public void LoadModel(string modelPath)
{
    _model = _mlContext.Model.Load(modelPath, out _);
}

```

60

```
}
```

```
private string RemoveLinks(string text)
```

```
{
```

```
    if (string.IsNullOrEmpty(text)) return text;
```

```
    var linkPattern = @"https?:\V\[\^s]+";
```

```
    return Regex.Replace(text, linkPattern, string.Empty);
```

```
}
```

```
private string ParseDate(object date)
```

```
{
```

```
    if (date is string dateString)
```

```
    {
```

```
        if (DateTime.TryParse(dateString, out DateTime parsedDate))
```

```
        {
```

```
            return parsedDate.ToString("yyyy-MM-dd");
```

```
        }
```

```
    else
```

```
    {
```

```
        return string.Empty;
```

```
    }
```

```
}
```

```
else
```

```
{
```

```
    return string.Empty;
```

```
}
```

```
}
```

```
public NewsPrediction Predict(NewsArticle article)
```

```
{
```

```
    if (_model == null)
```

```
    {
```

```
        throw new InvalidOperationException("Model is not loaded. Train or load the  
model before making predictions.");
```

```
    }
```

```
    var predictionEngine = _mlContext.Model.CreatePredictionEngine<TrainingData,  
NewsPrediction>(_model);
```

```

var data = new TrainingData
{
    Title = article.title,
    Content = RemoveLinks(article.text),
    Author = article.subject,
    PublicationDate = article.date.ToString()
};

var prediction = predictionEngine.Predict(data);

return prediction;
}
}

```

Файл GenericRepository

```

public class GenericRepository
{
    private readonly ConnectionFactory _cFactory;

    public GenericRepository(IConfiguration configuration)
    {
        var connectionString = configuration.GetConnectionString("DefaultConnection");
        _cFactory = new ConnectionFactory(connectionString);
    }

    public async Task<List<Dictionary<string, object>>> GetUsersAsync()
    {
        string query = "SELECT * FROM Users";
        return await _cFactory.ExecuteQueryAsync(query);
    }

    public Task<int> AddUserAsync(string name, string email, string lastName, string
password)
    {
        string hashedPassword = BCrypt.Net.BCrypt.HashPassword(password);

```

```

        string query = $"INSERT INTO Users (Name, Email, Last_Name, Password)
VALUES (@Name, @Email, @Last_Name, @Password)";
        var parameters = new Dictionary<string, object>
        {
            { "@Name", name },
            { "@Email", email },
            { "@Last_name", lastName },
            { "@Password", hashedPassword }
        };

        return _cFactory.ExecuteNonQueryAsync(query, parameters);
    }
}

```

Файл NewsFileProcessor

```

public class NewsFileProcessor
{
    public static List<NewsArticle> LoadNewsFromFile(string filePath)
    {
        var config = new CsvConfiguration(CultureInfo.InvariantCulture)
        {
            HasHeaderRecord = true,
        };

        using (var reader = new StreamReader(filePath))
        using (var csv = new CsvReader(reader, config))
        {
            var records = csv.GetRecords<NewsArticle>().ToList();

            foreach (var record in records)
            {
                record.title = RemoveLinks(record.title);
                record.text = RemoveLinks(record.text);
                record.subject = RemoveLinks(record.subject);
            }

            return records;
        }
    }
}

```

```

private static string RemoveLinks(string input)
{
    if (string.IsNullOrEmpty(input))
        return input;

    string urlPattern = @"https?:[/^\s]+";
    return Regex.Replace(input, urlPattern, string.Empty).Trim();
}
}

```

Файл Appsettings.json

```

{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",

  "ConnectionStrings": {
    "DefaultConnection": "Server=DESKTOP-IQDMGLR\\SQLEXPRESS;Database=MasterWork;Trusted_Connection=True;Connection Timeout=5;TrustServerCertificate=True;"
  },

  "Kestrel": {
    "Limits": {
      "MaxRequestBodySize": 209715200
    }
  }
}

```

Файл Program.cs

```

using Master_Work.Repository;
using Microsoft.AspNetCore.Http.Features;

var builder = WebApplication.CreateBuilder(args);

```

```
builder.Services.AddHttpContextAccessor();

builder.Services.AddControllersWithViews();
builder.Services.AddScoped<GenericRepository>();

builder.Services.Configure<FormOptions>(options =>
{
    options.MultipartBodyLengthLimit = 100_000_000; // 100 MB
});

builder.WebHost.ConfigureKestrel(options =>
{
    options.Limits.MaxRequestBodySize = 200_000_000; // 200 MB
});

builder.Services.AddSession(options =>
{
    options.IdleTimeout = TimeSpan.FromMinutes(30); // Set session timeout
    options.Cookie.HttpOnly = true; // Protect against XSS
    options.Cookie.IsEssential = true; // Required for GDPR compliance
});

var app = builder.Build();

if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    // The default HSTS value is 30 days. You may want to change this for production
    scenarios, see https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();
app.UseSession();
```

65

```
app.UseAuthorization();
```

```
app.MapControllerRoute(  
    name: "default",  
    pattern: "{controller=Home}/{action=Index}/{id?}");
```

```
app.Run();
```