МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інженерії
програмного забезпечення
_____ Євген ДАВИДЕНКО
«___»_____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

СИСТЕМА ВИЯВЛЕННЯ ФЕЙКОВИХ НОВИН

Спеціальність 121 Інженерія програмного забезпечення

Освітня програма «Інженерія програмного забезпечення»

Здобувач _____Андрій ГОНЧАР
«___»_____ 2024 р.

Керівник PhD, ст. викладач _____ Катерина АНТІПОВА
«___»_____ 2024 р.

Миколаїв – 2024

Чорноморський національний університет імені Петра Могили

(повне найменування закладу вищої освіти)

Факультет	Комп'ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Другий (магістерський)
Освітній ступень	Магістр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри інженерії
програмного забезпечення
_____Євген ДАВИДЕНКО
«____»_____2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

_____**Гончара Андрія Андрійовича**_____
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи
_____Система виявлення фейкових новин_____

Затверджена наказом ЧНУ ім. Петра Могили від «04» вересня 2024 р.
№___220__

2. Строк представлення кваліфікаційної роботи «___»_____2024 р.

3. Очікуваний результат – розроблена система.

4. Перелік питань, що підлягають розробці:

– дослідження різних підходів і моделей машинного навчання для задачі визначення фейкових новин, аналіз їх ефективності, переваг та недоліків;

– вивчення основних ознак, які можуть свідчити про неправдивість новинного контенту;

– моделювання архітектури та функціональних елементів системи;

– проєктування програмного забезпечення з урахуванням вимог;

– навчання моделі машинного навчання, оптимізованої для задачі класифікації новинного контенту;

– перевірка функціональності та оцінка точності системи на основі реальних новинних даних, аналіз результатів та визначення напрямків для вдосконалення.

5. Перелік графічних матеріалів – презентація

Керівник роботи	_____	Катерина АНТІПОВА
	Особистий підпис	*Власне ім'я ПРІЗВИЩЕ*
Здобувач	_____	Андрій ГОНЧАР___
	Особистий підпис	*Власне ім'я ПРІЗВИЩЕ*

Дата видачі завдання « ____»_____20_____р

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема: Система виявлення фейкових новин

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КМР	10.09.2024 р.	11.09.2024 р.	виконано
2.	Огляд літератури за темою роботи	13.09.2024 р.	19.09.2024 р.	виконано
3.	Складання календарного плану КМР	20.03.2024 р.	21.03.2024 р.	виконано
4.	Аналіз предметної області	24.09.2024 р.	26.09.2024 р.	виконано
5.	Розробка проєктних рішень	28.09.2024 р.	30.09.2024 р.	виконано
6.	Моделювання та конструювання ПЗ	01.10.2024 р.	06.10.2024 р.	виконано
7.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	11.10.2024 р.	07.11.2024 р.	виконано
8.	Оформлення КМР та презентації	10.11.2024 р.	20.11.2024 р.	виконано
9.	Відгук керівника КМР	27.11.2024 р.	27.11.2024 р.	виконано
10.	Попередній захист	28.11.2024 р.	28.11.2024 р.	виконано
11.	Завершення оформлення КМР та презентації	29.11.2024 р.	12.12.2024 р.	виконано
12.	Рецензування	13.12.2024 р.	13.12.2024 р.	виконано
13.	Захист кваліфікаційної роботи	19.12.2024 р.	19.12.2024 р.	виконано

Розробив здобувач Гончар А. А. _____

(прізвище, ім'я, по батькові) *(підпис)*

«__» _____ 2024 р.

Керівник роботи PhD, ст. викладач Антіпова К. О. _____

(посада, прізвище, ім'я, по батькові) *(підпис)*

«__» _____ 2024 р.

АНОТАЦІЯ

до кваліфікаційної магістерської роботи

«Система виявлення фейкових новин»

Здобувач 608м гр.: Гончар Андрій Андрійович

Керівник: PhD, ст. викладач Антіпова К. О.

В умовах глобальної цифровізації та стрімкого розвитку соціальних мереж проблема поширення фейкових новин набуває особливої актуальності. Дезінформація становить серйозну загрозу для суспільства, впливаючи на прийняття рішень та формування громадської думки, особливо під час кризових ситуацій та виборчих кампаній.

Об'єктом дослідження є процес автоматизованої верифікації інформаційного контенту.

Предметом дослідження виступають методологічні підходи та алгоритмічні рішення у сфері машинного навчання для багаторівневої класифікації текстового контенту.

Метою роботи є покращення точності розпізнавання фейків шляхом розробки системи виявлення фейкових новин на основі інтеграції передових алгоритмів глибокого машинного навчання та комп'ютерної лінгвістики.

Для досягнення поставленої цілі було визначено наступні завдання:

– дослідження різних підходів і моделей машинного навчання для задачі визначення фейкових новин, аналіз їх ефективності, переваг та недоліків;

– вивчення основних ознак, які можуть свідчити про неправдивість новинного контенту;

– моделювання архітектури та функціональних елементів системи;

– проєктування програмного забезпечення з урахуванням вимог;

– навчання моделі машинного навчання, оптимізованої для задачі класифікації новинного контенту;

перевірка функціональності та оцінка точності системи на основі реальних новинних даних, аналіз результатів та визначення напрямків для вдосконалення.

У першому розділі проведено ґрунтовний аналіз сучасних методів виявлення фейкових новин, досліджено актуальні рішення та системи-аналоги, сформульовано функціональні та нефункціональні вимоги до розроблюваної системи. Окрема увага приділена специфікації програмного забезпечення та визначенню ключових технічних характеристик.

Другий розділ присвячено детальному моделюванню системи з використанням UML-діаграм. Розроблено діаграми варіантів використання, що відображають взаємодію користувачів з системою, діаграми діяльності для візуалізації бізнес-процесів та діаграми послідовностей для демонстрації обміну повідомленнями між компонентами системи.

У третьому розділі представлено архітектурне рішення системи, обґрунтовано вибір технологій та інструментів розробки. Детально описано особливості обраної моделі DeBERTa-v3, її переваги для задачі класифікації текстів та процес оптимізації гіперпараметрів. Окремо розглянуто технологічний стек для серверної та клієнтської частин системи.

Четвертий розділ містить комплексний опис процесу реалізації системи, включаючи підготовку даних, навчання моделей машинного навчання, розробку серверної та клієнтської частин. Представлено результати порівняльного аналізу ефективності різних моделей та детальні результати модульного тестування, що підтверджують надійність та стабільність розробленого рішення.Робота містить 108 сторінок, 31 рисунків, 45 таблиці та 2 додатки. При написанні роботи використано 17 джерел.

Ключові слова: машинне навчання, фейкові новини, нейронні мережі, deberta-v3, обробка природної мови, класифікація тексту, мікросервісна архітектура.

ABSTRACT

of the Master's Thesis

«Fake news detection system»

Applicant of group 608m: Honchar Andrii Andriiovych

Supervisor: Ph.D., Senior Lecturer Antipova K. O.

In the context of global digitalization and rapid social media development, the problem of fake news proliferation becomes increasingly critical. Disinformation poses a serious threat to society, influencing decision-making and public opinion formation, especially during crises and election campaigns.

The object of research is the process of automated verification of information content.

The subject of research encompasses methodological approaches and algorithmic solutions in machine learning for multilevel text content classification.

The aim of the thesis is to improve fake news detection accuracy by developing a system based on the integration of advanced deep learning algorithms and computational linguistics.

To achieve this goal, the following tasks were defined:

– researching various approaches and machine learning models for the task of identifying fake news, analyzing their effectiveness, advantages, and disadvantages;

– studying the main features that may indicate the falsity of news content;

– modeling the architecture and functional elements of the system;

– designing software with consideration of requirements;

– training a machine learning model optimized for the task of classifying news content;

– testing the functionality and evaluating the accuracy of the system based on real news data, analyzing the results, and identifying directions for improvement.

The first chapter provides a thorough analysis of modern fake news detection methods, examines current solutions and analogous systems, and formulates functional and non-functional requirements for the system under development. Special attention is paid to software specifications and defining key technical characteristics.

The second chapter is devoted to detailed system modeling using UML diagrams. Use case diagrams illustrating user interaction with the system, activity diagrams for business process visualization, and sequence diagrams demonstrating message exchange between system components have been developed.

The third chapter presents the system's architectural solution and justifies the choice of technologies and development tools. The features of the chosen DeBERTa-v3 model, its advantages for text classification tasks, and the hyperparameter optimization process are described in detail. The technology stack for both server and client parts of the system is separately considered.

The fourth chapter contains a comprehensive description of the system implementation process, including data preparation, machine learning model training, and server and client-side development. The results of comparative analysis of different models' effectiveness and detailed unit testing results are presented, confirming the reliability and stability of the developed solution.

The work contains 108 pages, 31 figures, 45 tables, and 2 appendices. 17 sources were used in writing the thesis.

Keywords: machine learning, fake news, neural networks, deberta-v3, natural language processing, text classification, microservice architecture.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

КМР	–	кваліфікаційна магістерська робота
ОС	–	операційна система
ПЗ	–	програмне забезпечення
UML	–	unified modeling language
API	–	application programming interface
BERT	–	bidirectional encoder representations from transformers
CNN	–	convolutional neural network
CSS	–	cascading style sheets
DOM	–	document object model
GPU	–	graphics processing unit
HTML	–	hyper text markup language
JSON	–	javascript object notation
JWT	–	JSON web token
ML	–	machine learning
MLP	–	multi-layer perceptron
NB	–	naive bayes
NLP	–	natural language processing
ORM	–	object-relational mapping
REST	–	representational state transfer
RF	–	random forest
RNN	–	recurrent neural network
RTK	–	redux toolkit
SCSS	–	sassy CSS
SMOTE	–	synthetic minority over-sampling technique
SVM	–	support vector machine
TF-IDF	–	term frequency-inverse document frequency
URL	–	uniform resource locator

ВСТУП

У епоху глобальної цифровізації 2024 року інформаційно-комунікаційні технології трансформувалися у фундаментальний елемент повсякденної діяльності та соціальної взаємодії кожного члена суспільства, впливаючи на всі аспекти - від професійної діяльності до міжособистісних комунікацій. З розвитком інтернету, соціальних мереж та інших онлайн-ресурсів споживання інформації відбувається миттєво, що створює безпрецедентні можливості для поширення новин. Однак водночас виникає і серйозна проблема — поширення фейкових новин, які можуть маніпулювати свідомістю громадян, створювати паніку, дезінформувати або навіть шкодити окремим особам чи державам. Ця проблема стає все більш нагальною, оскільки доступ до засобів масової інформації значно спростився, а кількість споживачів онлайн-новин зростає щороку.

Особливо гостро ця проблема постала у часи глобальних криз, таких як пандемія COVID-19 та війни, коли дезінформація може негативно впливати на суспільні настрої, створювати паніку та навіть спонукати до небезпечних дій. Спостережувані тенденції підкреслюють критичну необхідність протидії дезінформації, що стало пріоритетним завданням для медіа-спільноти, державних інституцій та провідних технологічних корпорацій в умовах глобальної цифровізації. Проте, в умовах децентралізованих соціальних мереж та різноманітних онлайн-платформ, ручний моніторинг і перевірка всіх джерел є неможливими. Тому оптимальним рішенням стає впровадження передових технологій глибокого машинного навчання та спеціалізованих нейромережевих систем штучного інтелекту, які забезпечують автоматизований багаторівневий аналіз, семантичну класифікацію та верифікацію контенту в режимі реального часу.

Актуальність теми полягає в тому, що дезінформаційний контент та маніпулятивні медіаматеріали здійснюють масштабний вплив на різноманітні соціально-демографічні групи населення, формуючи викривлене сприйняття політичних, економічних, соціальних та культурних процесів. Вони можуть бути

інструментом інформаційних війн, порушення прав громадян, підриву довіри до медіа та демократичних процесів. Особливо небезпечною є поширеність фейкових новин під час виборчих кампаній, суспільних криз або пандемій, коли суспільство потребує достовірної інформації для ухвалення рішень.

Практична цінність цієї роботи полягає в розробці інтелектуальної програмної платформи з набором моделей машинного навчання для аналізу та верифікації контенту, що забезпечує автоматизовану ідентифікацію дезінформації з можливістю вибору оптимального алгоритму класифікації. Завдяки сучасним технологіям штучного інтелекту та машинного навчання з'явилася можливість створення алгоритмів, здатних аналізувати тексти новин, порівнювати їх із надійними джерелами, визначати типові патерни маніпуляцій та оцінювати достовірність контенту. Такі системи є важливим інструментом у боротьбі з дезінформацією, оскільки дозволяють суттєво підвищити рівень критичного споживання інформації, як серед журналістів, так і серед пересічних користувачів.

Розробка автоматизованих систем ідентифікації дезінформації в цифровому медіапросторі виступає об'єктом дослідження для міждисциплінарних наукових колективів, що включають фахівців у сферах інформаційних технологій, медіакомунікацій, соціології та юриспруденції. На даний момент розроблено кілька підходів до вирішення цієї проблеми, проте більшість з них ще потребують вдосконалення.

Головна складність полягає в тому, що фейкові новини постійно еволюціонують та підлаштовуються під нові обставини. Окрім цього, аналіз контенту потребує глибокого розуміння природної мови, що є серйозною технічною перешкодою.

Предметом дослідження виступають методологічні підходи та алгоритмічні рішення у сфері машинного навчання, що застосовуються для багаторівневої класифікації текстового контенту, з особливим фокусом на передові архітектури глибинного навчання та інструментарій комп'ютерної лінгвістики, які забезпечують автоматизовану детекцію дезінформаційних матеріалів у цифровому медіапросторі.

Метою роботи є покращення точності розпізнавання фейків шляхом розробки системи виявлення фейкових (неправдивих) новин, що базується на інтеграції передових алгоритмів глибокого автоматизованого навчання та комп'ютерної лінгвістики для обробки природномовних текстів. Для досягнення цієї мети було сформульовано наступні ключові завдання:

– дослідження різних підходів і моделей машинного навчання для задачі визначення фейкових новин, аналіз їх ефективності, переваг та недоліків;

– вивчення основних ознак, які можуть свідчити про неправдивість новинного контенту;

– моделювання архітектури та функціональних елементів системи;

– проєктування програмного забезпечення з урахуванням вимог;

– навчання моделі машинного навчання, оптимізованої для задачі класифікації новинного контенту;

– перевірка функціональності та оцінка точності системи на основі реальних новинних даних, аналіз результатів та визначення напрямків для вдосконалення.

Таким чином, кінцеві результати даної роботи можуть слугувати та використовуватися як базис для подальших наукових досліджень у галузі автоматизації виявлення дезінформації, а також для практичного застосування в медіаіндустрії та інших сферах, де важлива достовірність інформації.

Апробація результатів кваліфікаційної роботи. Основний зміст та висновки кваліфікаційної роботи викладені й обговорені на всеукраїнській науковій конференції Могилянські читання – 2024 (див. додаток А).

1 АНАЛІЗ СУЧАСНИХ МЕТОДІВ ІДЕНТИФІКАЦІЇ НЕДОСТОВІРНИХ НОВИН

1.1 Аналіз предметної області

Фейкові новини – це явище виникло разом із появою медіа, однак з розвитком цифрових технологій, Інтернету та соціальних мереж ця проблема набула всесвітніх масштабів, тому що вплив цих новин є міжконтинентальним[1]. Фейкові новини можна визначити як навмисне створені повідомлення, що мають на меті дезінформувати або ввести в оману аудиторію. Вони можуть поширювати неправдиву інформацію, перебільшувати факти або навмисно спотворювати дійсність для досягнення певних політичних, економічних чи соціальних цілей.

В інформаційному суспільстві, де доступ до новин став необмеженим, користувачі щоденно взаємодіють з величезною кількістю інформаційних повідомлень. Однак складність полягає в тому, що значна частина цих повідомлень може бути неточною або неправдивою. Це підриває довіру до традиційних медіа, які зобов’язані подавати перевірену інформацію. Крім того, сучасні користувачі Інтернету нерідко споживають інформацію з неперевірених джерел, таких як блоги, форуми або соціальні мережі, що ще більше ускладнює відділення правди від брехні.

Серед основних причин поширення фейкових новин виділяють наступні:

– політичний вплив, а саме те, що фейкові новини часто використовуються як інструмент пропаганди та маніпуляції під час виборчих кампаній, міжнародних конфліктів або внутрішніх політичних криз;

– економічні вигоди, деякі медіаорганізації створюють сенсаційний або маніпулятивний контент з метою залучення більшої кількості читачів та отримання прибутків від реклами;

– соціальні фактори тому, що часто фейкові новини використовуються для підсилення соціальної напруги або для формування негативного ставлення до певних груп людей, організацій або ідей.

Феномен фейкових новин привернув увагу науковців через серйозні наслідки, які вони можуть мати для суспільства. Фейкова інформація здатна спричинити соціальну дезінформацію, підривати довіру до офіційних джерел інформації, впливати на рішення виборців та створювати небезпеку для громадської безпеки.

Значну роль у поширенні фейкових новин відіграють соціальні мережі. У Facebook, Twitter, Instagram, TikTok та інших платформах інформація поширюється з величезною швидкістю, а алгоритми платформ зазвичай спрямовані на максимальну взаємодію користувачів. Це призводить до того, що сенсаційні або емоційно забарвлені повідомлення, які часто є фейковими, мають більшу ймовірність бути поширеними.

Однією з особливостей фейкових новин у соціальних мережах є віртуальні ехо-камери, коли користувачі бачать лише ті повідомлення, що підтверджують їхні вже існуючі переконання, формуючи своєрідний замкнутий цикл сприйняття інформації[2]. Це значно ускладнює боротьбу з фейковими новинами, оскільки користувачі можуть не тільки сприймати їх за правду, але й активно поширювати у своєму середовищі.

Поширення фейкових новин через соціальні мережі також активно використовується для інформаційних війн. Низка досліджень свідчить про те, що окремі країни та організації цілеспрямовано використовують фейкові новини для дискредитації політичних опонентів, створення напруженості в міжнародних відносинах або втручання у внутрішню політику інших держав.

Було помічено, що наукові дискусії про «дезінформацію та соціальні медіа» почали з'являтися в дослідженнях після 2008 року, це можна побачити в академічних інформаційних системах, наприклад, Google Scholar, де спостерігається експоненціальне зростання числа наукових публікацій та дослідницьких робіт з проблематики інформаційної безпеки та протидії дезінформації після цього періоду. Пізніше, у 2010 році, ця тема привернула більше уваги, коли були використані боти у Твіттері або поширювалися фейкові новини про заміну сенатора США. У цей період одночасно зростали кампанії ненависті та

активність фейкових підписників. Як видно з рисунку 1, на якому показано кількість статей про дезінформацію, опублікованих між 2005 і 2021 роками в трьох базах даних: Scopus, Springer та EBSCO, академічна активність щодо дезінформації, схоже, набула більшого імпульсу після президентських виборів у США 2016 року, коли соціальні медіа-платформи, вочевидь, вплинули на вибори[3].

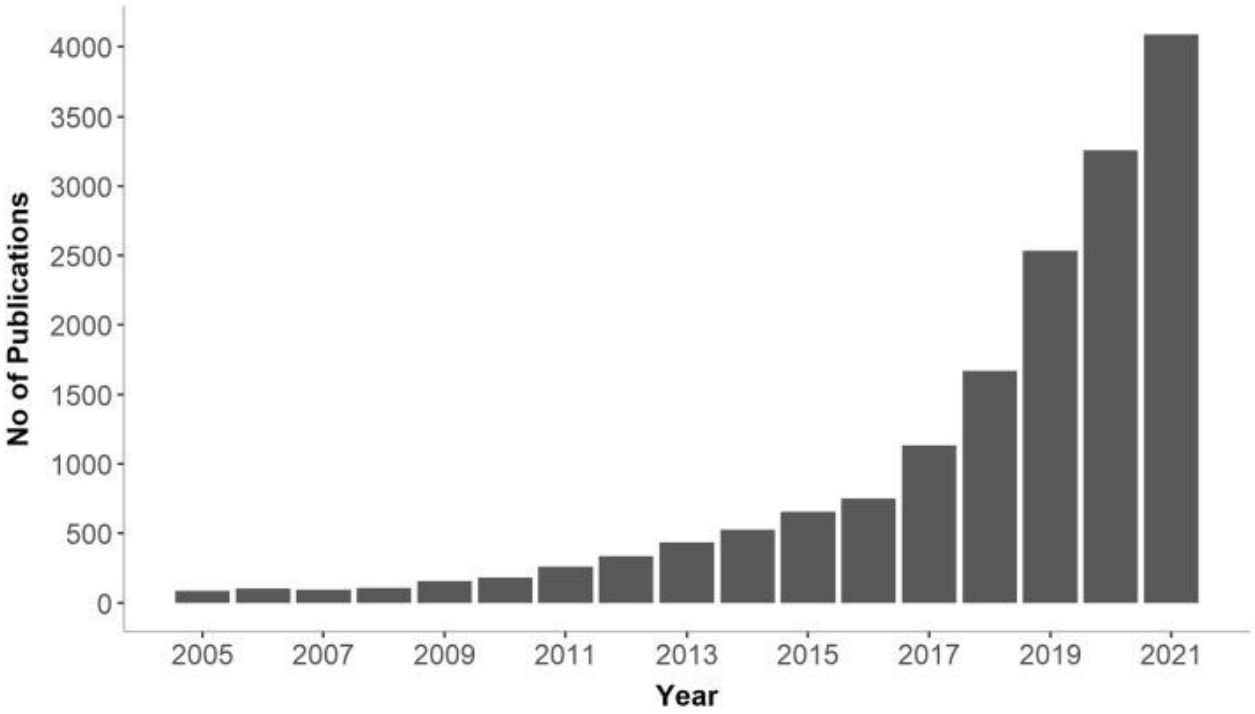


Рисунок 1.1 – Діаграма кількості публікування фейкових новин

Для виявлення фейкових новин у наукових дослідженнях використовується кілька підходів, які зазвичай поділяються на три основні групи: лінгвістичні методи, контент-аналіз та методи, засновані на аналізі джерел та поширення інформації.

Лінгвістичні методи базуються на аналізі мовних характеристик тексту. Це може бути морфологічний, синтаксичний або семантичний аналіз, а також дослідження тональності. Ідея полягає в тому, що тексти фейкових новин мають певні характерні риси, такі як використання емоційно забарвлених слів, суперлативів або ненормативної лексики. Одним із таких підходів є аналіз тональності, що дозволяє оцінити емоційне навантаження новинного повідомлення. Фейкові новини часто мають дуже поляризовану або негативну

тональність, що є однією з ознак маніпуляції[4]. Також може використовуватися стилометрія – метод, який аналізує стиль написання тексту, враховуючи такі параметри, як довжина речень, частота використання окремих частин мови та повторювані граматичні структури.

Контент-аналіз передбачає дослідження змісту новин з метою виявлення відповідності інформації реальним подіям, науковим фактам або статистичним даним. Один з поширених підходів — фактчекінг, який дозволяє порівняти інформацію, представлену у новині, з офіційними джерелами або достовірними фактами.

Методи, засновані на аналізі джерел та поширення інформації, досліджують соціальні мережі, блоги та інші онлайн-платформи, де можуть поширюватися фейкові новини. Одним із найважливіших аспектів цього підходу є аналіз мереж поширення інформації. Наприклад, якщо певна новина демонструє аномальні патерни розповсюдження через нелегітимні облікові записи або автоматизовані системи дистрибуції контенту, це свідчить про високу ймовірність присутності дезінформаційних елементів.

Також використовується аналіз довіри до джерел. Новини, які поширюються від ненадійних або анонімних джерел, мають більшу ймовірність бути неправдивими. Використання методу перевірки надійності джерел дозволяє відфільтрувати неперевірену інформацію та мінімізувати ризики дезінформації.

Сучасні системи детекції дезінформації досягли революційного прогресу через імплементацію предиктивної аналітики та нейрокогнітивних обчислень. Інтеграція передових алгоритмів статистичного навчання та багатошарових нейромережевих архітектур уможливила створення високоефективних інструментів автоматизованої верифікації інформаційного контенту.

Моделі машинного навчання заслуговують на одну із провідних ролей в вирішенні проблеми аналізу тексту новин. Вони використовуються для навчання системи виявленні фейкових новин на основі великих обсягів даних, та виявляти приховані закономірності, що можуть бути недоступними для звичайного аналізу. Основна мета таких моделей полягає в автоматичному виявленні фейкових новин,

що ґрунтується на аналізі лексичних, стилістичних та структурних особливостей тексту.

Моделі машинного навчання працюють за принципом навчання на основі даних (даних, що містять як правдиві, так і неправдиві новини). Для цього використовується супервізоване навчання, коли модель отримує вхідні дані (текст новин) разом із відповідними мітками (фейкові або правдиві новини). В процесі навчання модель виявляє закономірності, що відрізняють фейкові новини від правдивих.

Після завершення етапу навчання модель може застосовуватися для аналізу нових текстів і класифікації їх як фейкових або правдивих. В основі цього процесу лежить виявлення певних лінгвістичних ознак, стилістичних маркерів, використання мови емоцій або надмірного вживання сенсаційної лексики, що часто властиво неправдивим новинам.

Також ML-системи можуть адаптуватися до нових фейкових новин і вдосконалювати свої алгоритми завдяки постійному навчанню на нових даних. З часом моделі можуть ставати точнішими, оскільки вони «вчаться» на основі нових патернів, що з'являються у медійному середовищі.

Моделі машинного навчання демонструють високу обчислювальну ефективність при паралельній обробці та аналізі масштабних інформаційних потоків у режимі реального часу, що набуває критичного значення в сучасну епоху цифрових медіа, де спостерігається експоненціальне зростання обсягу новинного контенту.

1.2. Аналіз аналогів

Після переглядання систем виявлення фейкових новин, було виділено п’ять аналогів, а саме: Snopes, FactCheck.org, Hoaxy, BERT.

Першим для аналізу було взято платформу Snopes, користувацький інтерфейс та основні елементи головної сторінки якої візуально представлені на рисунку 1.2 для демонстрації ключових компонентів системи.

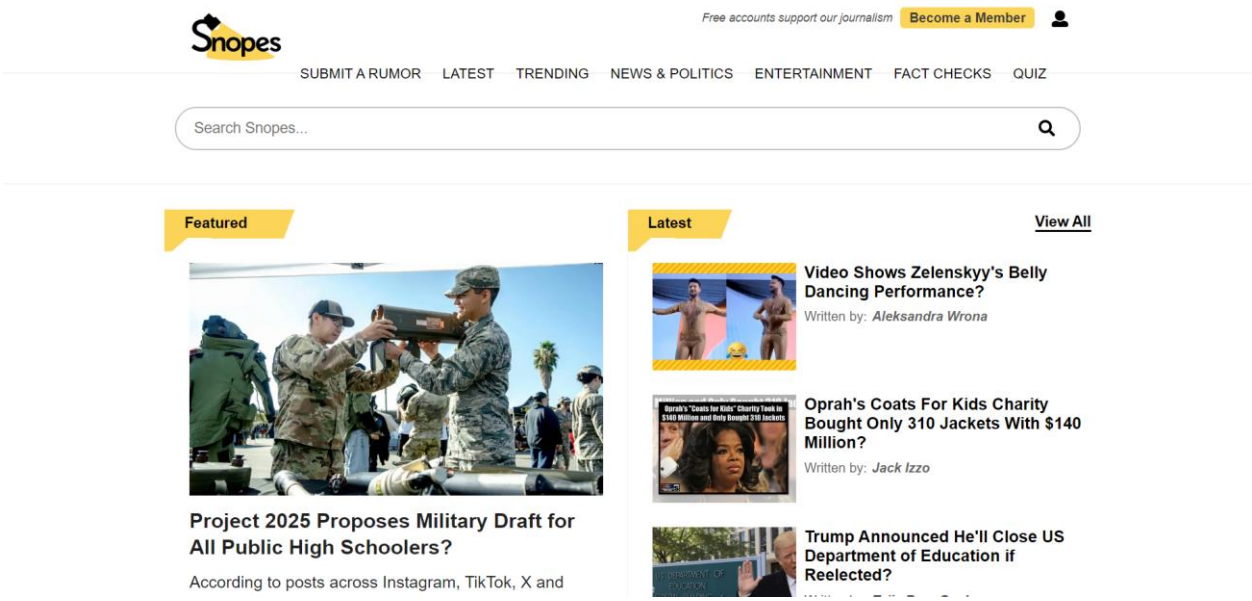


Рисунок 1.2 – Візуалізація користувацького інтерфейсу та функціональних елементів головної сторінки платформи Snopes

Snopes – це один із найвідоміших фактчекінгових сайтів, заснований ще в 1994 році, який спеціалізується на виявленні фейкових новин і дезінформації[5]. Snopes проводить ручну перевірку новин, розглядаючи контент з різних джерел, а також надаючи детальну інформацію про перевірені факти. Ретельний аналіз цієї системи був зведений в таблиці 1.1.

Таблиця 1.1 – Характеристика гри Snopes

Назва характеристики	Опис
Назва	Snopes
Тип системи	Ручна перевірка фактів
Переваги	Висока точність перевірки новин через те, що вся інформація проходить через людей експертів в області ідентифікування новин на фейк. Досвід та авторитет сервісу роблять його надійним джерелом фактчекінгу.

Кінець таблиці 1.1

Перелік функціоналу	Основний функціонал: – ручна перевірка фактів, а саме кожна новина або інформація перевіряється експертами вручну; – аналіз різних джерел за допомогою збирання інформації з різних джерел, включаючи офіційні документи, наукові дослідження, урядові дані для більш точного результату; – категоризація новин (правдива, частково правдива, фейкова); – публікування результатів з поясненням, чому новина вважається правдивою або неправдивою.
Недоліки	Оскільки перевірка фактів виконується вручну, цей підхід важко масштабувати для аналізу великої кількості новин, а також процес ручного фактчекінгу може займати час, що є недоліком у світі, де інформація поширюється дуже швидко.

Наступним було розглянуто та проаналізовано аналог FactCheck.org, також було представлено головну сторінку системи (рисунок 1.3).

Рисунок 1.3 – Інтерфейс основної сторінки платформи FactCheck.org

FactCheck.org – ще один відомий фактчекінговий ресурс, який займається верифікацією політичних заяв та новин[6]. Він орієнтований переважно на

політичні новини та заяви в США і працює на основі аналізу відкритих джерел та урядової статистики. Деталізований аналіз було сформовано в таблиці 1.2.

Таблиця 1.2 – Характеристика гри FactCheck.org

Назва характеристики	Опис
Назва	FactCheck.org
Тип системи	Ручна перевірка фактів
Переваги	Спеціалізація на політичному контенті робить FactCheck.org потужним інструментом для виявлення неправдивих заяв у політиці, додатково, програмний застосунок має велику кількість кваліфікованих аналітиків та експертів.
Перелік функціоналу	Основний функціонал: – аналіз політичних новин і заяв, можна побачити, що основна увага приділяється політичним темам, таким як заяви політиків, урядові рішення, економічна політика; – ручний збір інформації, як і з Snopes (див. таблицю 1.1) експерти вручну перевіряють дані на основі офіційних джерел та відкритих статистичних даних; – публікація детальних звітів, а саме кожен фактчекінг супроводжується детальним аналізом і поясненням того, чому та чи інша заява є правдивою або неправдивою; – оцінка достовірності, це означає, що результати публікуються із висновками про рівень правдивості новини або заяви (правдива, маніпулятивна, фейкова тощо).

Кінець таблиці 1.2

Недоліки	Обмежена сфера дії, тому що система орієнтована на політичну тематику, тому не є універсальним інструментом для виявлення фейкових новин з інших сфер, до того ж низька автоматизація, обґрунтовується це тим, що FactCheck.org залежить від ручної перевірки, що знижує можливість швидкої обробки великого обсягу інформації.

Далі було досліджено та проаналізовано альтернативу Hoaxy, а також зображено головну сторінку (рисунок 1.4).

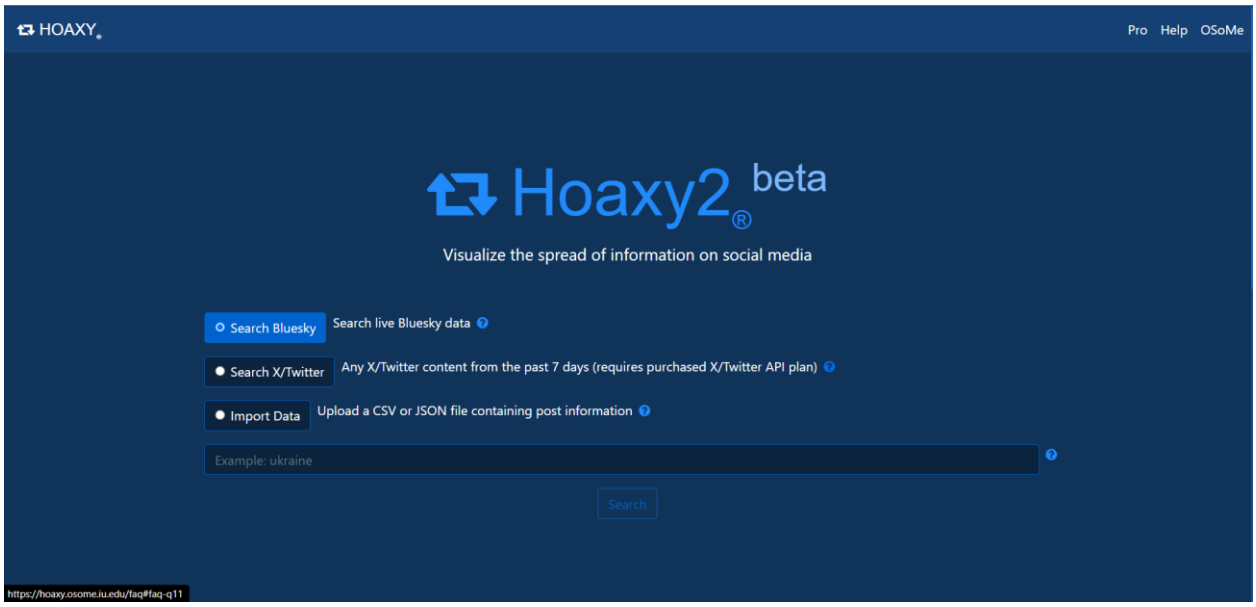


Рисунок 1.4 – Основний функціонал та головна сторінка Hoaxy

Hoaxy – це інструмент для візуалізації поширення фейкових новин і перевірених новин у соціальних медіа[7]. Він дозволяє користувачам бачити, як певна новина або чутка поширюється через різні соціальні мережі. Hoaxy використовує алгоритми для аналізу поширення новин і може виявляти боти, що сприяють розповсюдженню неправдивої інформації. Більш детальний опис було наведено в таблиці 1.3.

Таблиця 1.3 – Характеристика гри Ноаху

Назва характеристики	Опис
Назва	Ноаху
Тип системи	Інструмент візуалізації
Переваги	Візуалізація поширення новин, а саме Ноаху дає користувачам змогу відстежувати, як поширюється новина в інтернеті, що може допомогти визначити її джерело та патерни поширення, також система проводить аналіз ботів, тому що Ноаху має інструменти для виявлення ботів, які часто використовуються для розповсюдження фейкових новин.
Перелік функціоналу	Основний функціонал: – візуалізація поширення новин, Ноаху надає графічні візуалізації того, як новини поширюються через соціальні медіа; – проведення аналізу взаємодій у соціальних мережах, а саме відстежує, які користувачі діляться новинами, які облікові записи поширюють новини (люди чи боти) – виявлення ботів, використання алгоритмів для визначення ботів, що можуть бути джерелами або каталізаторами поширення фейкових новин; – пошук за ключовими словами, користувачі можуть вводити ключові слова або URL-адреси для відстеження конкретних новин – аналіз ретвітів і цитувань, Ноаху показує, скільки разів новина була ретвітнута або процитована, що може бути індикатором її вірусності.

Кінець таблиці 1.3

Недоліки	Не виявляє фейковість безпосередньо, тому що Ноаху сам по собі не визначає, чи є новина фейковою, а лише показує, як вона поширюється, тому оцінка достовірності новин залишається на користувачеві. Також складність інтерпретації даних, що робить користування складною для звичайного користувача, тому що інструмент надає велику кількість візуалізованих даних, що в свою чергу ускладнює розуміння без належної підготовки.

Окрім традиційних фактчекінгових платформ, сучасні дослідження активно впроваджують штучний інтелект у процес виявлення фейкових новин, саме тому було проаналізовано систему, яку сформовану на основі глибокого навчання, таку як Poe, тому що вона теж може бути використана для аналізу контенту новин, а головну сторінку системи продемонстровано на рисунку 1.5.

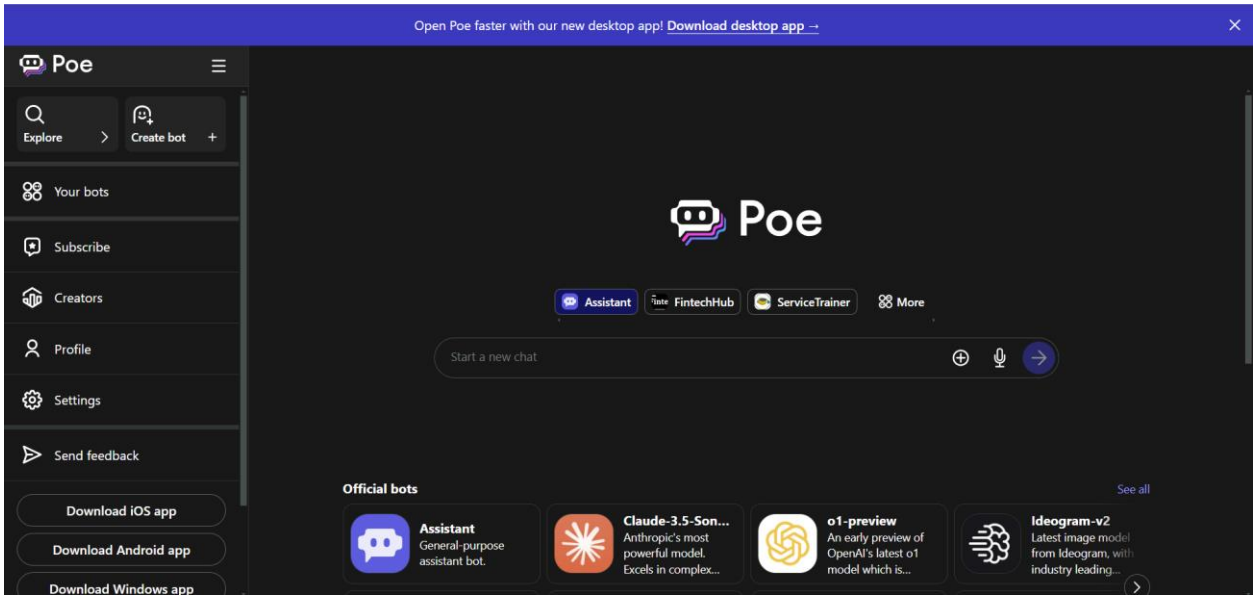


Рисунок 1.5 – Інтерактивний інтерфейс платформи Poe з візуалізацією елементів керування штучним інтелектом

Poe – це платформа для взаємодії з різними AI-моделями, зокрема такими, як GPT-4 від OpenAI та Claude від Anthropic. Вона дозволяє користувачам ставити запитання і отримувати відповіді від різних ШІ-моделей через інтерфейс чату[8]. Платформа створена компанією Quora і надає можливість швидкого доступу до кількох моделей для розв'язання різноманітних задач.

Таблиця 1.4 – Характеристика гри Poe

Назва характеристики	Опис
Назва	Poe
Тип системи	Автоматизовані модель ШІ
Переваги	Доступ до потужних ШІ-моделей, система надає доступ до мовних моделей, які можна використовувати для автоматичного аналізу тексту, виявлення патернів фейкових новин або маніпуляцій, а можливість швидкої обробки тексту надає перевагу при перевірці великої кількості новин на правдивість або маніпулятивність через швидкий аналіз великих текстових наборів.
Перелік функціоналу	Основний функціонал: – мультимодельна платформа, Poe дозволяє вибирати різні моделі ШІ для конкретних задач, користувач може обрати GPT-4 для генерації тексту або Claude для складніших аналітичних завдань; – платформа пропонує зручний інтерфейс для інтерактивного спілкування з ШІ, де користувач може формулювати запити у форматі діалогу; – аналіз та генерація тексту, система підтримує інструменти для створення тексту, аналізу даних, перекладу та іншого використання мовних моделей; – Інтеграція з Quora.

Кінець таблиці 1.4

Недоліки	Залежність від навчальних даних, тому що як і всі мовні моделі, Poe залежить від якості даних, на яких була навчена, а якщо модель не була тренована на конкретних даних про фейкові новини або маніпуляції, це може вплинути на її ефективність, також обмеження у розумінні контексту, цей недолік випливає з першого, тому що завдяки тренуванню моделі можна досягти гарного розуміння контексту, а через недостатнє тренування ШІ-моделі не завжди здатні повністю зрозуміти соціальний або культурний контекст новин, що може призвести до помилок у виявленні фейків.

Також було сформовано таблицю 1.5, яка демонструє дані для кращого порівняння з аналогами.

Таблиця 1.5 – Порівняння з аналогами

Функціонал	КМР	Snopes	FactCheck.org	Hoaxy	Poe	Пояснення
Аналізу тексту новин	+	+	+	-	+	КМР, Snopes, FactCheck.org та Poe аналізують текст новин, тоді як Hoaxy фокусується на візуалізації поширення новин, а не на детальному аналізі змісту.
Автоматизація	+	-	-	-	+	Автоматизація передбачає автоматичну перевірку новини на правдивість
Навчання на підготовленому набору даних	+	-	-	-	-	Тренування моделі на спеціально підготовлених датасетах фейкових і правдивих новин, забезпечуючи кращу точність. Poe може робити більше помилок через відсутність цього етапу. Snopes і FactCheck.org потребують ручної перевірки, що уповільнює процес.
Інтерактивна робота з користувачами	+	-	-	-	+	Надавання зручного інтерфейсу для інтерактивної роботи з ШІ

1.3 Специфікація вимог до програмного забезпечення

В представленій структурі технічної документації наведено формалізовану специфікацію функціональних та системних вимог до розроблюваного програмного забезпечення[9].

МЕТА ТА ОБСЯГ ПРОЄКТУ

Призначення системи (додатка), для якої створюється програмне забезпечення

Програмна платформа з інтегрованими алгоритмами штучного інтелекту призначена для автоматичного виявлення та аналізу фейкових новин, з метою підвищення точності інформації та запобігання дезінформації в медіапросторі.

Узгодження, затверджені в програмній документації

Згідно з схваленою технічною специфікацією програмного забезпечення було погоджено використовувати мову програмування Python та бібліотеки TensorFlow, Scikit-learn, Transformers, Keras для реалізації алгоритмів аналізу даних, а також Django, TypeScript та React.

Обмеження проєкту програмного забезпечення

Остаточний термін завершення розробки програмного забезпечення заплановано на перший квартал 2024 року.

ЗАГАЛЬНИЙ ОПИС

Область використання

Ця платформа розроблена для медіасервісів, інформаційних агентств, соціальних мереж та інших онлайнресурсів, де необхідна перевірка автентичності інформації.

Характеристика користувачів

Технічні та кваліфікаційні вимоги до кінцевих користувачів системи включають: наявність обчислювального пристрою (персональний комп'ютер, смартфон або ноутбук) з стабільним підключенням до глобальної всесвітньої мережі Інтернет, додатково, мати базові навички роботи з програмним забезпеченням та веб-інтерфейсами.

Загальні обмеження

Жодних обмежень не передбачено.

ФУНКЦІЇ СИСТЕМИ

Автоматичне виявлення фейкових новин на основі текстового контенту.

Опис функції

Інноваційна функціональність спрямована на ефективне викриття дезінформаційного контенту шляхом комплексного аналізу текстового матеріалу. Ця передова технологія застосовує прогресивні алгоритми машинного навчання для ретельної оцінки надійності відомостей на основі комплексного набору даних для навчання.

ВИМОГИ ДО ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

Вхідна і вихідна інформація

Вхідна інформація приходить в форматі звичайного тексту, після перевірки користувач отримує вихідну інформацію у вигляді результату аналізу.

Вимоги до способів організації, збереження та ведення інформації

Інформація має зберігатися в структурованому сховищі даних реляційного типу, що забезпечить впорядкований доступ і оперативне виконання запитів. Вхідні дані повинні зберігатися у форматі JSON для легкості обробки та передачі між компонентами системи.

ВИМОГИ ДО ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

Для забезпечення високопродуктивної обробки даних необхідно застосовувати потужний процесор, який має як мінімум чотири обчислювальні ядра.

Оперативна пам'ять: не менше 8 ГБ для швидкої обробки запитів і зберігання даних у пам’яті.

Місце на диску: щонайменше 500 ГБ на SSD для швидкого доступу до бази даних та зберігання текстових даних.

ОС: Windows, macOS або дистрибутиви Linux (наприклад, Ubuntu або CentOS) для забезпечення сумісності з бібліотеками Python.

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Архітектура програмної системи

Структура системи має ґрунтуватися на архітектурі клієнт-сервер, де клієнтська частина відповідає за взаємодію з користувачами, а серверна компонента займається обробкою запитів, зберіганням даних та реалізацією алгоритмів.

Системне ПЗ

Для забезпечення гнучкості розгортання, система має підтримувати широкий спектр операційних систем, зокрема Windows, macOS та Linux.

Бази даних: використання реляційних баз даних для організації зберігання даних та їх обробки.

Системи управління версіями: використання Git для контролю версій коду та спільної роботи над проєктом.

Мережне програмне забезпечення

Підтримка стандартних протоколів, таких як HTTP/HTTPS для безпечного обміну даними між клієнтською та серверною частинами системи.

Мова і технологія розробки ПЗ

Python (версія 3.7 або вище) для розробки основної логіки програмного забезпечення, завдяки її потужним бібліотекам для нейромережевих обчислень та лінгвістичного аналізу.

TypeScript для розробки інтерфейсів для зручної взаємодії системи і користувача.

Інтерфейс користувача

Інтуїтивно зрозумілий вебінтерфейс, що дозволяє користувачам взаємодіяти з системою через браузер. Він повинен бути адаптивним і доступним на різних пристроях (десктопи, планшети, смартфони).

Апаратний інтерфейс

Апаратним інтерфейсом є комп'ютер, ноутбук чи смартфон з вільним доступом до інтернету.

Програмний інтерфейс

Реалізація RESTful API для взаємодії між клієнтською та серверною частинами системи.

Комунікаційний протокол

Використання протоколів HTTP та HTTPS для забезпечення безпечного та ефективного обміну даними між клієнтською та серверною частинами.

Використання JSON для обміну даними між компонентами системи, що забезпечує простоту та зрозумілість структури даних.

ВЛАСТИВОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Доступність

Рішення має бути постійно доступним для користувачів, за наявності інтернет-з'єднання.

Вебінтерфейс має бути адаптований для користувачів з різними можливостями.

Супроводжуваність

Система не потребує супроводжуваності.

Переносимість

Система повинна забезпечувати кросплатформну сумісність з гетерогенними операційними середовищами, включаючи Windows, macOS та Unix-подібні системи, та пропонувати автоматизований процес розгортання для безшовної інсталяції на різноманітних апаратних конфігураціях.

Продуктивність

Система має гарантувати швидку обробку запитів та аналіз новин з найменшими можливими затримками.

Надійність

Система спроєктована для забезпечення автономного функціонування без необхідності постійного моніторингу та супроводу.

Безпека

Архітектура системи реалізована таким чином, що не потребує додаткових механізмів забезпечення надійності, оскільки використовує вбудовані засоби

відмовостійкості фреймворків Django та React, а також нативні механізми збереження даних в PostgreSQL.

ІНШІ ВИМОГИ

Усі необхідні вимоги вже повністю сформульовані та детально описані вище; додаткових уточнень чи доповнень не потребується.

Висновки до розділу 1

Перший розділ містить ґрунтовний системний аналіз вибраної предметної області. Було детально розкрито об’єкт та предмет дослідження, що дозволило зрозуміти важливість розробки ефективних систем для аналізу та виявлення недостовірної інформації. Також описано структурні і функціональні особливості існуючих рішень, що підкреслює різноманітність методів та підходів для реалізації систем у даній сфері.

Підсумовуючи, можна зрозуміти, що сучасний стан інформаційних технологій вказує на тенденції розвитку і необхідність впровадження нових підходів, зокрема на основі машинного навчання та штучного інтелекту.

2 МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Створення варіантів використання

Для більш зрозумілого опису сценаріїв спочатку було створено рівні доступу для користувачів, які було імплементовані в нашу систему, а саме було виділено два рівня користувача:

– неавторизований користувач;

– авторизований користувач.

Неавторизований користувач – це користувач, який не виконав реєстрацію і немає власного облікового запису для авторизації в системі або має власний обліковий запис, але не авторизувався. Для цього рівня користувачів система надає форму авторизації та можливість створити власний обліковий запис через спеціальний реєстраційний модуль, перевірка новин недоступна без проходження процедури авторизації.

Авторизований користувач – це користувач, який має обліковий запис в системі та він пройшов процес авторизації. Даний тип користувача має доступ на перевірку новин на фейк.

Далі було описано сценарії, як саме користувачі взаємодіють з системою, а також які кроки йому потрібно виконати, щоб досягти бажаного результату. Для описання сценаріїв було використано методологію UseCase, такий підхід дозволяє сфокусуватися на реальних потребах користувача та сприяє інтуїтивно зрозумілого інтерфейсу і покращенню загального досвіду використання.

Коротка форма написання сценарію (UseCase**) для задачі «Авторизація користувача»:**

Авторизація дозволяє користувачу отримати доступ до функціоналу перевірки новин. Для цього користувач надає свої логін і пароль, після чого система перевіряє їх на коректність. Якщо дані вірні, користувач отримує доступ до функцій системи.

Поверхнева форма написання сценарію (UseCase**) для задачі «Авторизація користувача»:**

Успішний сценарій

Користувач повинен ввести унікальну електронну пошту разом з паролем для входу в систему. Система перевіряє введені дані на відповідність з тими, що збережені в базі. У разі після успішної авторизації користувач отримує можливість користуватися функціоналом перевірки новин.

Альтернативні сценарії

Якщо користувач ввів дані невірно, то йому надається спроба ввести їх, а також відображається, що дані введено невірно.

Повна форма написання сценарію (UseCase**) для задачі «Авторизація користувача» (див. табл. 2.1).**

Таблиця 2.1 – UC-1 сценарій експлуатації

Назва	Авторизація користувача
Опис	Процес, за яким користувач входить у систему, щоб отримати доступ до функціоналу перевірки новин.
Учасники	Неавторизований користувач
Передумови	Користувач повинен мати обліковий запис у системі.
Постумови	Після успішної авторизації користувач отримує можливість скористатися функціоналом перевірки новин.
Основний сценарій	Користувач переходить на сторінку авторизації та вводить свою унікальну електрону пошту і пароль у відповідні поля. Після підтвердження дії система здійснює верифікацію введених даних. Якщо інформація введена правильно, користувач успішно авторизується та перенаправляється на головну сторінку, де отримує доступ до всіх функціональних можливостей.
Розширення сценарію	Якщо дані невірні, система виводить повідомлення про некоректне введення даних і надає можливість повторити спробу.

Коротка форма написання сценарію (UseCase**) для задачі «Реєстрація користувача»:**

Процес реєстрації надає новим користувачам можливість створити особистий обліковий запис для подальшого доступу до системи. Під час реєстрації користувач вводить необхідну інформацію, а саме ім’я, конфіденційний пароль та унікальну електронну пошта. Після успішного завершення реєстрації користувач

може увійти в платформу та використовувати її розширені функціональні можливості.

Поверхнева форма написання сценарію (UseCase**) для задачі «Реєстрація користувача»:**

Успішний сценарій

Користувач перенаправляється до розділу реєстрації, де вводить свої особисті дані, такі як ім’я, пароль довжиною більше 2 символів та унікальна електронна пошта. Після введення даних система перевіряє їх на відповідність і унікальність, зокрема перевіряє, чи не використовується email раніше. Якщо всі дані вірні, система створює обліковий запис в системі, а користувача перенаправляє на форму авторизації для подальших дій в системі.

Альтернативні сценарії

Якщо користувач вводить некоректні дані або паролі не збігаються, система демонструє спеціалізоване сповіщення про помилку і закликає виправити виявлені неточності. Якщо електронна адреса вже зареєстрована в системі, користувач отримує повідомлення про зайнятість цієї адреси та пропонується ввести інший email.

Повна форма написання сценарію (UseCase**) для задачі «Реєстрація користувача» (див. табл. 2.2).**

Таблиця 2.2 – Сценарій експлуатації UC-2

Назва	Реєстрація користувача
Опис	Процес ініціалізації нової облікової ідентифікації для подальшого доступу до платформи
Учасники	Суб'єкт, який не має попередньої авторизації.
Передумови	Суб'єкт, який не має попередньої авторизації, переходить на сторінку реєстрації
Постумови	Користувач успішно зареєстрований і може авторизуватися в системі.
Основний сценарій	Новий користувач ініціює процес реєстрації, заповнюючи поля з власним ім'ям, електронною адресою, паролем та його підтвердженням на спеціальній сторінці. Комплексна система, своєю чергою, здійснює перевірку введених даних на коректність, і за умови їх відповідності, спрямовує авторизованого суб'єкта на форму входу в обліковий запис.

Кінець таблиці 2.2

Розширення сценарію	Однак, у разі виявлення невідповідностей в наданих даних або невідповідності паролів, система відображає відповідне повідомлення про помилку. Крім того, якщо введена електронна адреса вже зареєстрована в платформі, вона повідомляє про це користувача та пропонує ввести альтернативний email.

Коротка форма написання сценарію (UseCase**) для задачі «Перевірка новини на фейковість»:**

Авторизований користувач вводить текст новини для перевірки. Система обробляє інформацію за допомогою алгоритмів і повертає результат правдива новина або ж навпаки.

Поверхнева форма написання сценарію (UseCase**) для задачі «Перевірка новини на фейковість»:**

Успішний сценарій

Авторизований користувач ініціює перевірку достовірності новини, для чого переходить на спеціалізовану сторінку. На цій сторінці він вводить текст новини, який система отримує і аналізує за допомогою алгоритмів машинного навчання. Після завершення комплексного аналізу, користувач отримує результат, що містить висновок про те, чи дана новина є фейковою, чи повністю правдивою.

Альтернативні сценарії

Якщо новина не може бути перевірена через брак даних, система виводить повідомлення про неможливість перевірки із пропозицією спробувати пізніше. Втім, за умов недоступності системи до серверних потужностей через технічні перешкоди чи збої в роботі, система відображає користувачу релевантне повідомлення про помилку з рекомендацією щодо повторної спроби звернення до функціоналу у подальшому.

Повна форма написання сценарію (UseCase**) для задачі «Перевірка новини на фейковість» (див. табл. 2.3).**

Таблиця 2.3 – Сценарій експлуатації UC-3

Назва	Верифікація новини на предмет правдивості
Опис	Процес перевірки новини на достовірність шляхом введення тексту новини.
Учасники	Автентифікований учасник
Передумови	Для доступу до функціональності платформи, особа має пройти процедуру авторизації та підтвердити свою ідентифікацію в якості зареєстрованого учасника.
Постумови	Суб'єкт отримує висновок щодо достовірності проаналізованої новини.
Основний сценарій	Авторизований учасник ініціює процедуру верифікації новинного матеріалу, для чого переходить на спеціалізовану сторінку. На даній сторінці він вводить заголовок та текст новинного повідомлення, яке комплексна система аналізує за допомогою алгоритмів машинного навчання. Після завершення детального опрацювання, суб'єкту надається результат перевірки, що містить висновок щодо ймовірності фейковості новини.
Розширення сценарію	У випадку, коли система не може здійснити повноцінний аналіз новини через недостатність вхідних даних, вона відображає повідомлення про неможливість верифікації. Крім того, за умов недоступності серверних потужностей, платформа надає релевантну інформацію про технічну помилку та рекомендує користувачу повторити спробу перевірки за деякий час.

Для полегшення розуміння основних функціональних аспектів системи та взаємодії користувачів з програмним забезпеченням, було створено діаграму варіантів використання[10].

Застосування діаграми варіантів використання дозволяє чітко окреслити ролі різних категорій користувачів, що, своєю чергою, сприяє визначенню функціональних можливостей, які система надаватиме кожній з цих груп. Додатково, діаграма сценаріїв допомогає показати взаємозв’язки між процесами, а саме використання механізмів include та extend дозволяє деталізувати кроки процесів, що допомагає розробникам та користувачам зрозуміти, як кожен етап взаємодіє з іншими. Окрім ефективного розподілу ролей, діаграма варіантів використання також уможливлює детальний опис можливих помилок та альтернативних сценаріїв, демонструючи, яким чином система реагує на

некоректні вхідні дані, надаючи користувачам можливість відновити або повторити процес, що, своєю чергою, забезпечує підвищення зручності в експлуатації.

Графічна схема варіантів використання даної системи було наведено на рисунку 2.1.

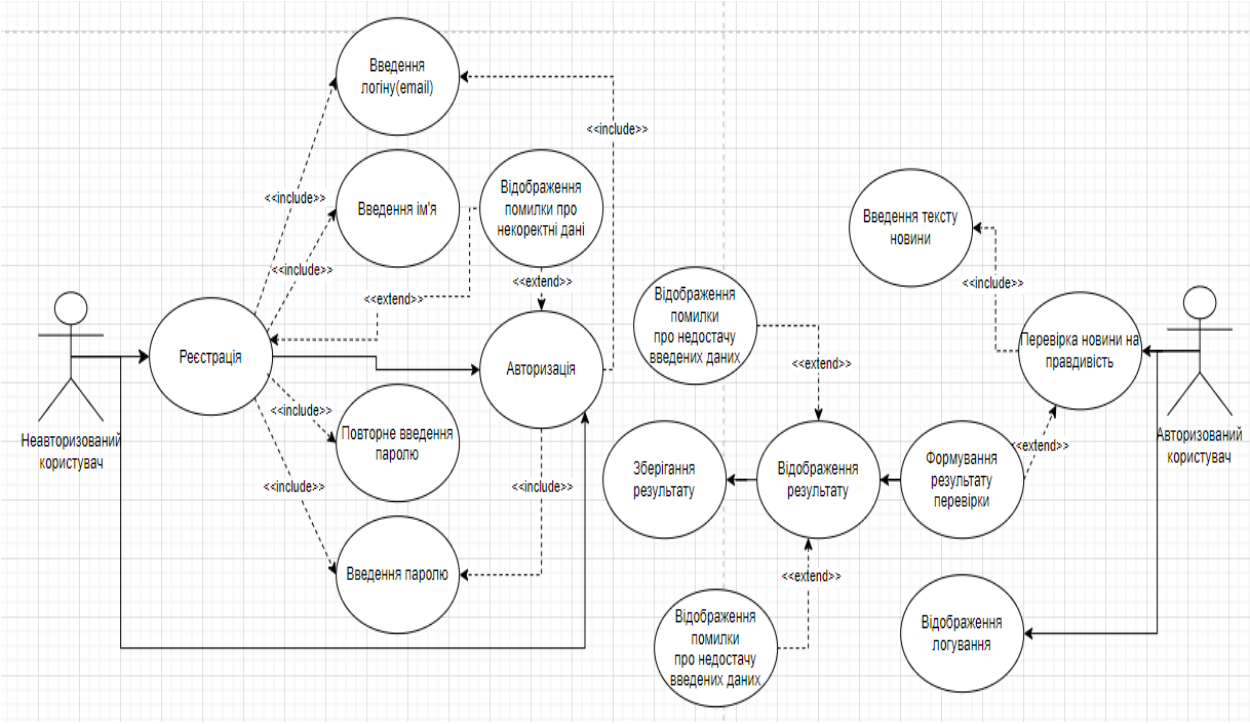


Рисунок 2.1 – Деталізована діаграма сценаріїв використання системи

Аналітичний опис діаграми варіантів використання з детальними поясненнями:

1. актори:

– неавторизований користувач – це користувач, який не має попередньо пройденої авторизації, його можливості обмежені лише реєстрацією, він не має доступу до перевірки новин до моменту авторизації;

– авторизований користувач – це користувач, який пройшов процедуру авторизації і має доступ до головної функції системи – перевірки новини на правдивість;

2. Сценарії використання (Use Cases):

– Процес створення облікового запису - це крок, який виконує неавторизований користувач для ініціалізації нового профілю. Цей процес містить кілька обов'язкових кроків: введення логіну (email), введення імені, введення пароля та його підтвердження. Якщо введені дані є некоректними або відсутні необхідні поля, система відображає відповідні повідомлення про помилки, після чого користувач може внести виправлення;

– авторизація – після реєстрації користувач може авторизуватися, ввівши логін і пароль, система перевіряє коректність введених даних і, у випадку успіху, надає доступ до основних функцій, якщо ж дані некоректні або недостатні, користувач отримає відповідне сповіщення;

– верифікація новини на предмет правдивості – це головна функція системи, доступна лише для авторизованих користувачів, користувач вводить текст новини, який система перевіряє за допомогою алгоритмів аналізу тексту, результат перевірки відображається для користувача, і він може містити інформацію про те, чи є новина правдивою чи фейковою, якщо система не може здійснити перевірку (наприклад, через недостатність даних), користувач також отримує відповідне повідомлення;

– відображення логування – відображення авторизованому користувачеві кількості перевірених новин, кількості правдивих та фейкових новин, тощо;

3. механізми розширення та включення (extend і include):

– механізми розширення та включення продемонстровані на діаграмі та описані вище в поясненнях до процесів.

Діаграма чітко структурує всі кроки взаємодії користувача з системою. Завдяки використанню механізмів include та extend, можна відслідкувати як обов'язкові етапи, так і можливі виняткові ситуації, що виникають при неправильному введенні даних або інших помилках. Це допоможе в подальшій розробці зрозуміти, які компоненти системи мають бути реалізовані та як вони мають взаємодіяти, також це зручно для документування всіх процесів і подальшого тестування системи.

2.2 Побудова діаграми діяльності (Activity diagram)

Діаграма діяльності – це тип діаграми в UML, який використовується для моделювання процесів або робочих потоків у системі[11]. Вона фокусується на послідовності дій або кроків, що виконуються в процесі. Діаграми діяльності є зручним засобом для візуалізації динаміки виконання функцій у системі та взаємодії між різними етапами процесу. Цей засіб є потужним інструментом для моделювання як стратегічних бізнес-процесів, так і детальних алгоритмічних рішень у програмному забезпеченні.

Діаграма діяльності показує кроки або дії, що виконуються одна за одною в процесі або функції. Кожна дія зображається як прямокутник із заокругленими кутами. Також діаграма описує, як дії пов'язані між собою. Стрілки вказують напрямок потоку від однієї дії до іншої, показуючи послідовність виконання, і включає вузли рішення, де визначається подальший шлях виконання процесу на основі певної умови. Вони відображаються у вигляді ромба. Застосовуючи діаграму діяльності, можна візуалізувати паралельні процеси або дії, що виконуються одночасно. Для цього використовуються спеціальні символи, які називаються розділення/злиття. Початкові та кінцеві стани позначаються чорним заповненим колом (початковий стан) та обведеним чорним колом (кінцевий стан). Вони показують, з чого починається процес і чим він завершується.

На рисунку 2.2 зображено діаграму діяльності для представлення покрокового сценарію взаємодії користувача з системою, включаючи процеси або рішення, що приймаються на основі певних умов.

На даній діаграмі діяльності проілюстровано процеси взаємодії користувача із системою в контексті реєстрації, авторизації та перевірки новин на фейковість. Діаграма включає три основні компоненти: користувача, клієнтську частину системи та серверну частину з базою даних.

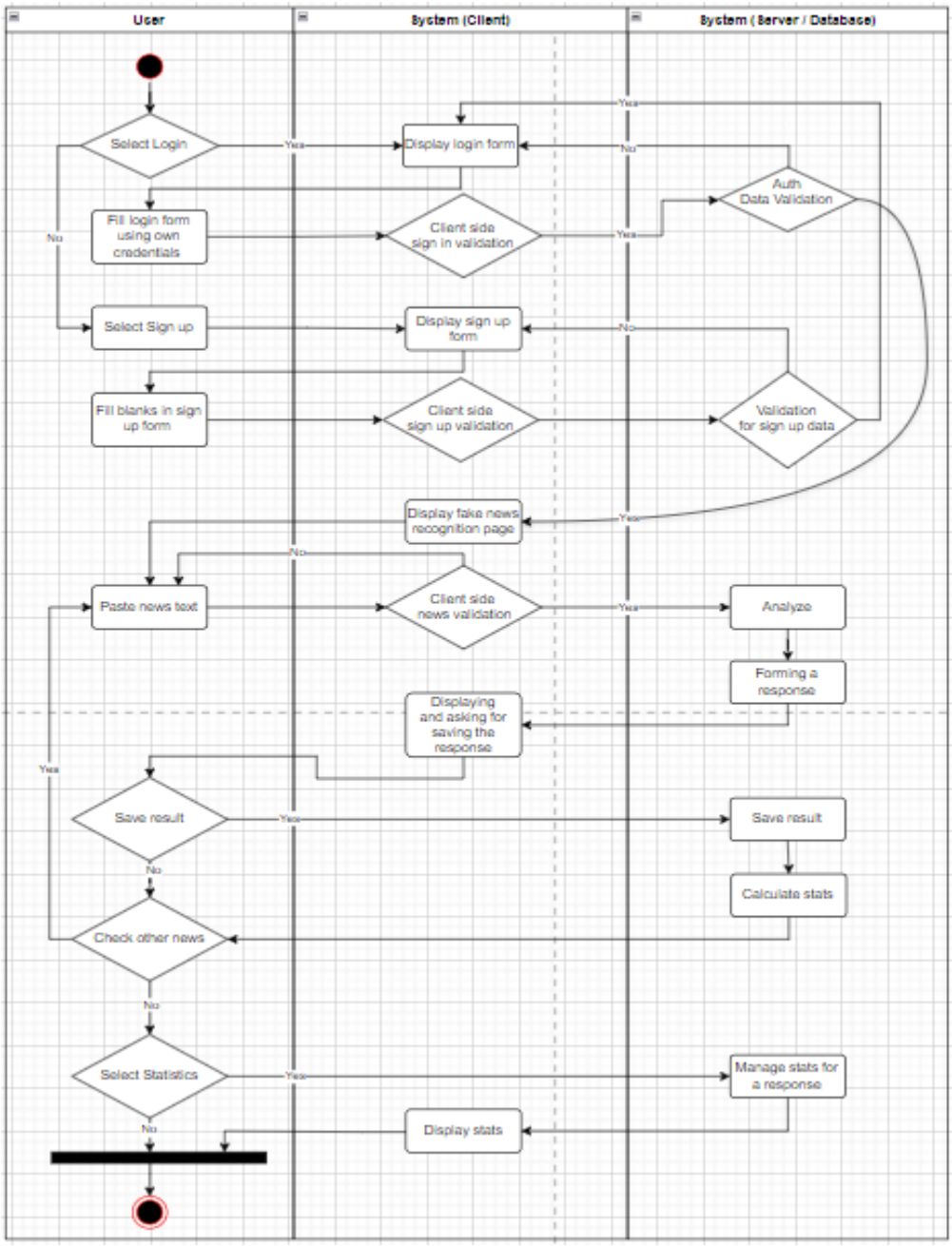


Рисунок 2.2 – Діаграма діяльності системи

На початку користувач здійснює вибір між авторизацією або реєстрацією. У випадку авторизації, він вводить свої дані, які перевіряються спочатку на клієнтській стороні, а потім передаються на сервер для валідації. Якщо обрано реєстрацію, користувач заповнює необхідні поля у формі. Ці дані також перевіряються на стороні клієнта для основних помилок, а потім надсилаються на сервер, де відбувається глибша валідація. Після успішного входу в систему переходить до сторінки перевірки новин.

Коли користувач потрапляє на сторінку перевірки новин, він вводить текст новини, яку хоче перевірити на правдивість. Ці дані також перевіряються на стороні клієнта перед тим, як відправитися на сервер. Серверна частина аналізує новину і повертає результат перевірки. Якщо новина є фейковою або правдивою, сервер формує відповідь і передає її клієнту, після чого клієнтська частина відображає результат на екрані користувача, також сервер може сформувати відповідь у вигляді помилки.

Після отримання результату користувач може продовжити перевірку інших новин або завершити сесію. Діаграма також демонструє моменти, коли виникають помилки, наприклад, у випадку некоректних даних при реєстрації або авторизації, і тоді система відображає повідомлення про помилку з відповідними інструкціями щодо виправлення.

Таким чином, діаграма діяльності наочно показує всі ключові етапи взаємодії користувача із системою та відображає, як система обробляє інформацію на клієнтській і серверній частинах для досягнення перевірки новини на фейковість.

2.3 Побудова діаграми послідовностей (Sequence Diagram)

Діаграма послідовностей — це перш за все один із видів UML-діаграм, який використовується для моделювання поведінки системи[12]. Головними задачами цієї діаграми полягає в розкритті, як об'єкти системи взаємодіють між собою у певній послідовності для виконання певного функціоналу. На діаграмі можуть бути представлені різні елементи системи та їх обмін повідомленнями у часовій шкалі.

Головною особливістю такої діаграми є те, що вона фокусується на хронології взаємодій між об'єктами, а це в свою чергу говорить про те, що кожен об'єкт або учасник має свою життєву лінію, яка демонструє, коли об'єкт активний і як він реагує на виклики від інших об'єктів. Важливо також те, що діаграма зображує обмін інформації у вигляді стрілок, які вказують на дії, що викликаються або на відповіді, які надходять в результаті. Це дозволяє побачити, де можуть виникати затримки або проблеми, які об'єкти можуть бути перевантажені, та зрозуміти, як удосконалити роботу системи, саме тому було створено діаграму

послідовностей для системи виявлення фейкових новин та продемонстровано на рисунку 2.3-2.4.

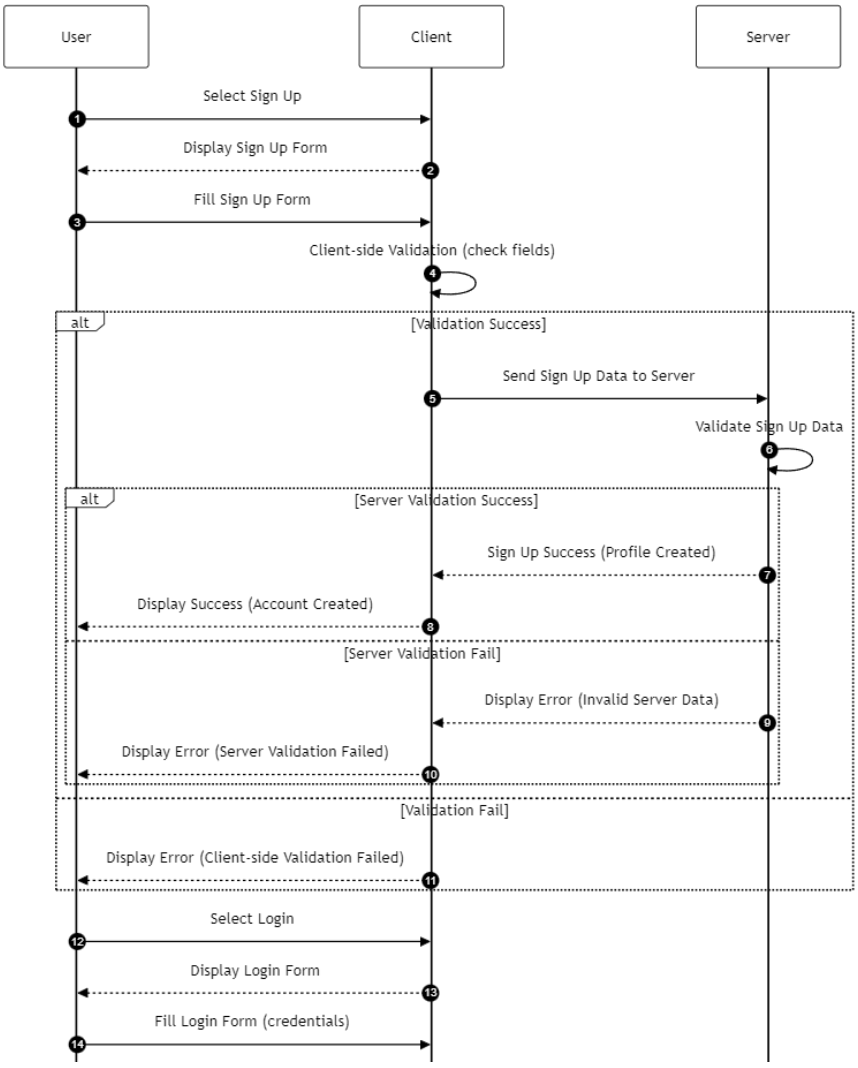


Рисунок 2.3 – Діаграма послідовностей системи

Дана діаграма візуально відображає послідовність взаємодій між різними учасниками системи під час виконання процесу. У цьому випадку діаграма послідовностей зосереджена на взаємодії користувача із клієнтською системою, яка, у свою чергу, спілкується з сервером, який підв'язаний до бази даних для виконання основних функцій.

По-перше, процес входу в систему починається з того, що користувач обирає опцію входу. Клієнтська система відображає форму для введення логіну та пароля. Користувач вводить свої облікові дані, після чого клієнтська частина системи проводить первинну валідацію.

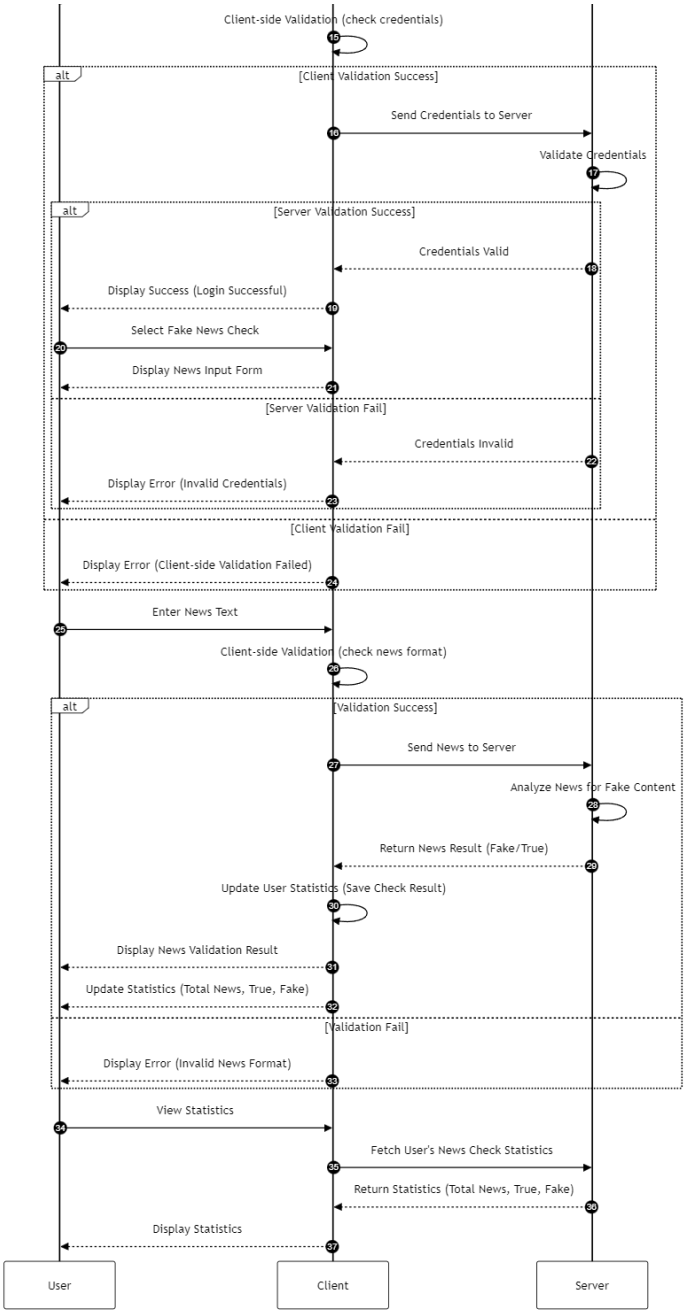


Рисунок 2.4 – Продовження діаграми послідовностей системи

Якщо введені дані відповідають вимогам, система відправляє їх на сервер для перевірки достовірності. Серверна частина перевіряє облікові дані, у випадку успішної аутентифікації, відповідає клієнту. У результаті клієнт відображає відповідну сторінку, на якій користувач може перевіряти новини. Якщо ж користувач вирішує зареєструватися, він обирає опцію реєстрації, і клієнтська частина системи відображає форму для введення даних. Користувач заповнює всі поля, а система перевіряє їх на правильність введення на стороні клієнтської частини системи. Якщо всі поля заповнені правильно, дані відправляються на

сервер для створення нового облікового запису. Після успішної валідації на сервері користувач отримує підтвердження про створення профілю та може авторизуватися.

Після входу в систему користувач може перейти до процесу перевірки новин. Для цього він вставляє текст новини, яку хоче перевірити. Клієнтська система знову перевіряє, чи введений текст відповідає вимогам, якщо валідація пройшла успішно, новина надсилається на сервер, де здійснюється аналіз на предмет фейковості. Результат перевірки повертається на клієнтську частину, яка відображає його користувачеві.

Висновки до розділу 2

У другому розділі було розглянуто ключові етапи проєктування та аспекти розробки системи для перевірки новин на фейковість. Ми дослідили основні сценарії використання системи, включаючи процес реєстрації, авторизації, та перевірки новин. Детально опрацьовано поведінкові моделі взаємодії (Use Case), вони забезпечили кристалізацію розуміння між користувачем і системою, а також розуміння ключового функціоналу можливостей, необхідних для її впровадження. Побудовані діаграми, зокрема діаграми діяльності та послідовностей, забезпечують ясність щодо внутрішньої логіки системи та підходів до її реалізації. Зокрема, діаграми відображають такі критичні моменти, як валідація даних, обробка помилок, а також аналіз новин на фейковість.

Крім того, важливим аспектом стало використання мокапів для візуалізації інтерфейсу та оцінки користувацького досвіду

Таким чином, дані підходи допомогли виявити можливі помилки на етапі проєктування та покращити розуміння логіки взаємодії між клієнтською і серверною частинами. Це дозволяє гарантувати відповідність кінцевого продукту потребам користувачів і спрощує процес розробки.

3 АРХІТЕКТУРА, МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

Для реалізації системи виявлення фейкових новин було обрано клієнт-серверну архітектуру, яка забезпечує оптимальний розподіл обчислювального навантаження та надає можливість ефективного масштабування[13]. Дана архітектура дозволяє централізовано зберігати та обробляти дані на сервері, в той час як клієнтська частина відповідає за взаємодію з користувачем та відображення результатів. На рисунку 3.1 представлено загальну схему архітектури системи, що відображає основні компоненти та їх взаємозв'язки.

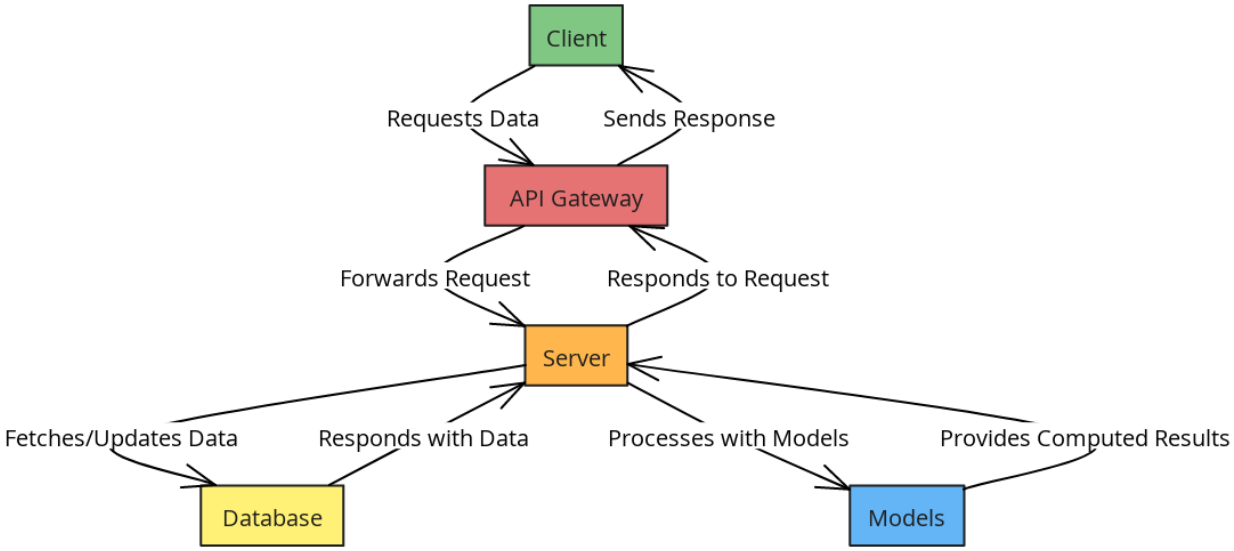


Рисунок 3.1 – Діаграма клієнт-серверної архітектури

Клієнтська частина системи реалізована у вигляді вебзастосунку, що забезпечує зручний інтерфейс для взаємодії користувачів з системою. Основним завданням клієнтської частини є надання користувачеві можливості вводити текст новини для аналізу та переглядати результати перевірки. Інтерфейс розроблено з урахуванням принципів user experience design, що робить взаємодію з системою інтуїтивно зрозумілою навіть для недосвідчених користувачів[14]. Важливою функцією клієнтської частини є первинна валідація введених даних, що дозволяє зменшити навантаження на сервер та покращити якість вхідних даних для аналізу.

На рисунку 3.2 представлено діаграму компонентів клієнтської частини, що демонструє структуру та взаємозв'язки між компонентами вебінтерфейсу системи.

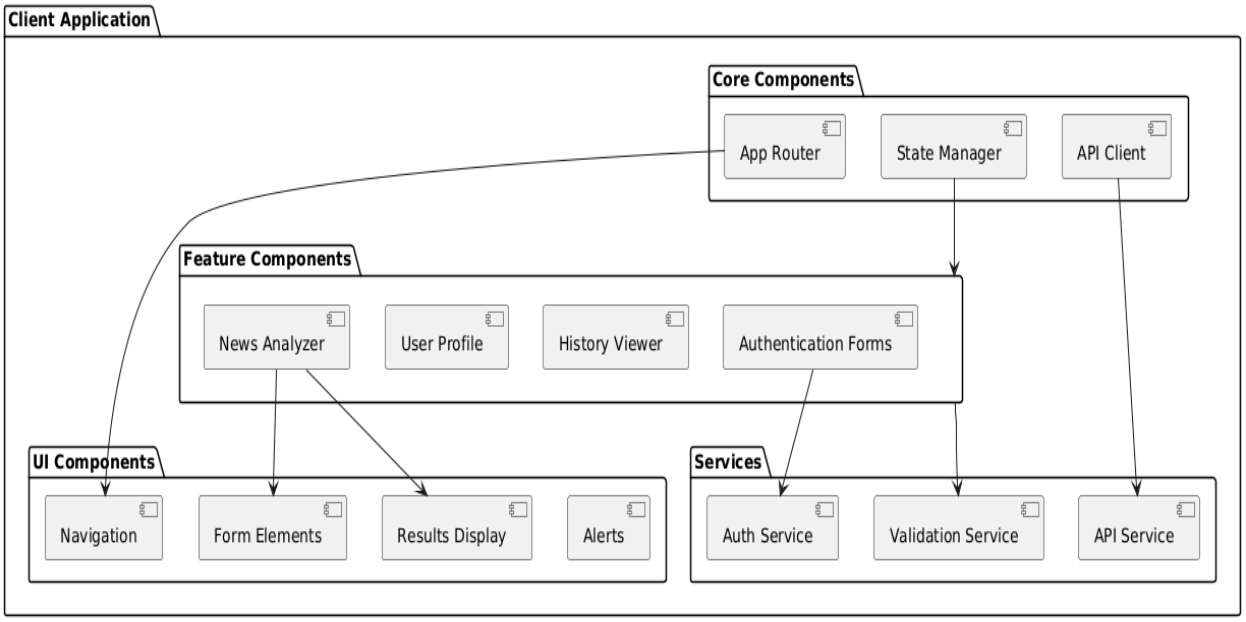


Рисунок 3.2 – Діаграма компонентів клієнтської частини

Серверна частина системи представляє собою складний комплекс взаємопов'язаних компонентів, кожен з яких відповідає за виконання специфічних завдань. Центральним елементом серверної частини є API Gateway, який виступає єдиною точкою входу для всіх клієнтських запитів. API Gateway забезпечує маршрутизацію запитів до відповідних мікросервісів, здійснює балансування навантаження та реалізує базові механізми безпеки.

За обробку та аналіз новин відповідає спеціалізований сервіс, який використовує потужності штучного інтелекту для виявлення фейкової інформації. Цей сервіс тісно взаємодіє з компонентом машинного навчання, де розгорнуті попередньо навчені моделі для класифікації текстів. Важливою особливістю архітектури є можливість горизонтального масштабування саме цього компоненту, оскільки він виконує найбільш ресурсомісткі операції. На рисунку 3.3 відображено діаграму взаємодії серверних компонентів, що показує як різні частини backend системи комунікують між собою.

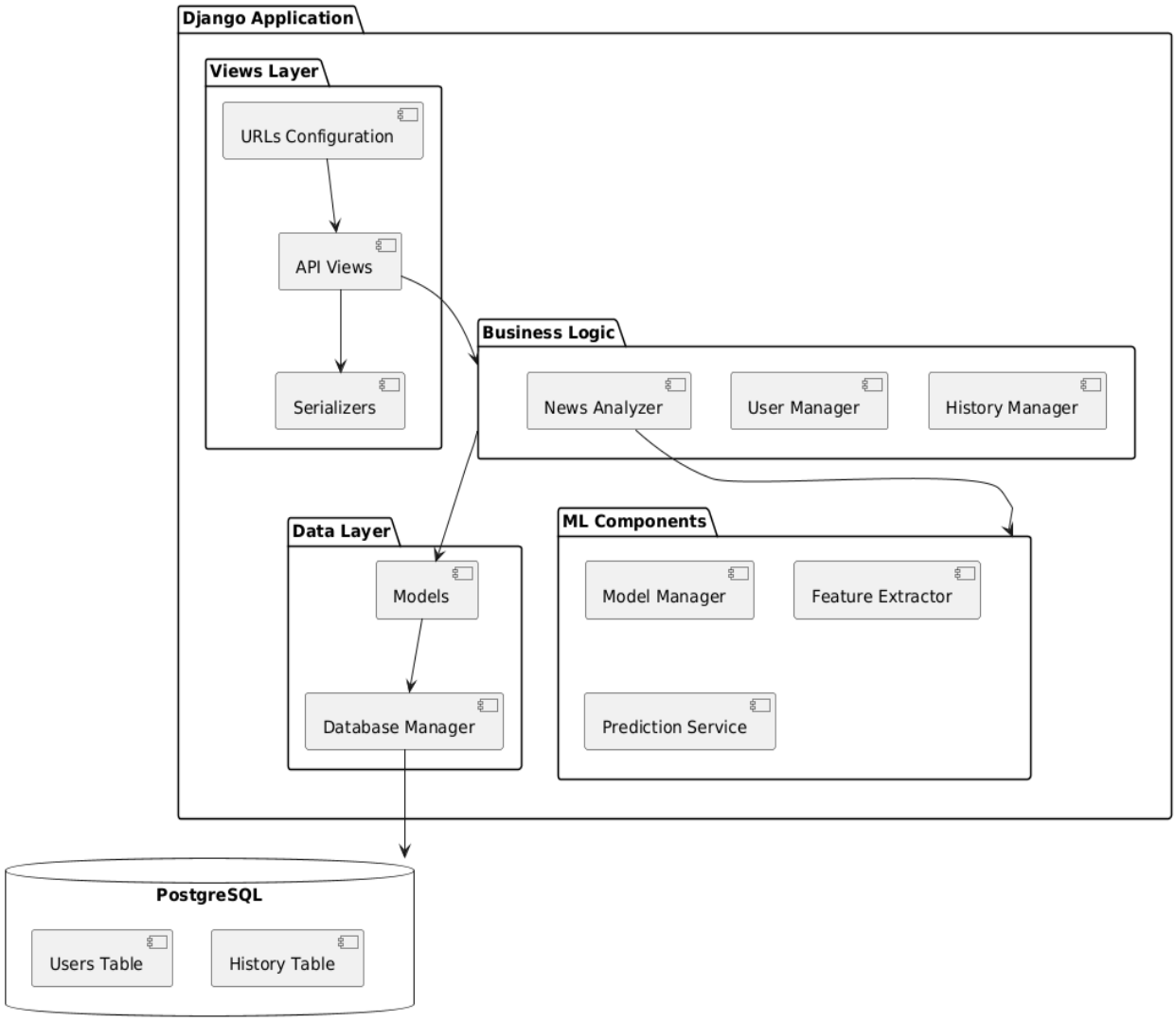


Рисунок 3.3 – Діаграма взаємодії серверних компонентів

Особливу увагу в архітектурі приділено безпеці даних та захисту від несанкціонованого доступу. Система аreутентифікації побудована на основі JWT токенів, що забезпечує надійну ідентифікацію користувачів та захист від підміни запитів[15].

Для персистування даних імплементовано високопродуктивне реляційне сховище PostgreSQL, що реалізує відмовостійке зберігання користувацьких профілів, аналітичних артефактів та результатів предиктивної класифікації[16]. Архітектура інтегрує механізми горизонтального масштабування реплікаційних нод для забезпечення безперебійної доступності та оптимізації розподілу навантаження при операціях зчитування. Для більш зручного розуміння, як саме

влаштована база даних, продемонстровано діаграму «сутність – зв'язок» на рисунку 3.4.

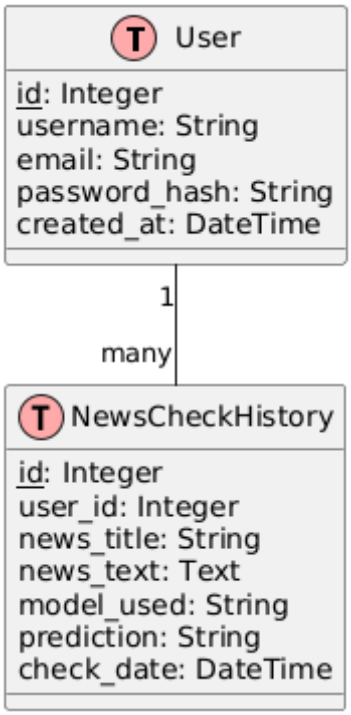


Рисунок 3.4 – Діаграма-модель структурної топології даних

Обрана архітектура також передбачає можливість подальшого розвитку системи та додавання нових функціональних можливостей. Модульна структура та чітко визначені інтерфейси взаємодії між компонентами дозволяють легко інтегрувати нові сервіси без необхідності суттєвої перебудови існуючої системи.

Таким чином, реалізована клієнт-серверна архітектура забезпечує надійну, масштабовану та безпечну роботу системи виявлення фейкових новин, надаючи користувачам зручний інструмент для перевірки достовірності інформації. Також на рисунку 3.5 візуалізовано карту інформаційних потоків в екосистемі, яка ілюструє шлях даних від моменту створення запиту до отримання результату.

Для реалізації серверної частини системи виявлення фейкових новин було обрано мікросервісну архітектуру, що забезпечує оптимальне рішення для розгортання, масштабування та підтримки системи. Головною перевагою такого підходу є можливість незалежного масштабування окремих компонентів системи відповідно до навантаження.

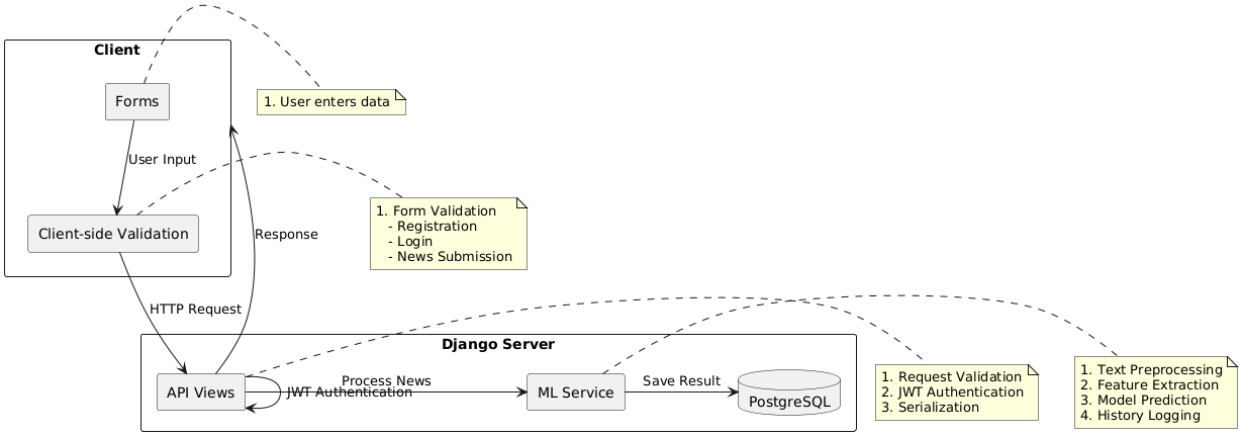


Рисунок 3.5 – Карта інформаційних потоків в екосистемі

Система складається з трьох основних мікросервісів: Authentication Service, ML Service та History Service, кожен з яких відповідає за окрему бізнес-логіку та може бути розгорнутий і масштабований незалежно від інших. Така декомпозиція забезпечує інтелектуальне масштабування та раціоналізацію споживання обчислювальних потужностей, функціонування мікросервісної архітектури системи зображено на рисунку 3.6.

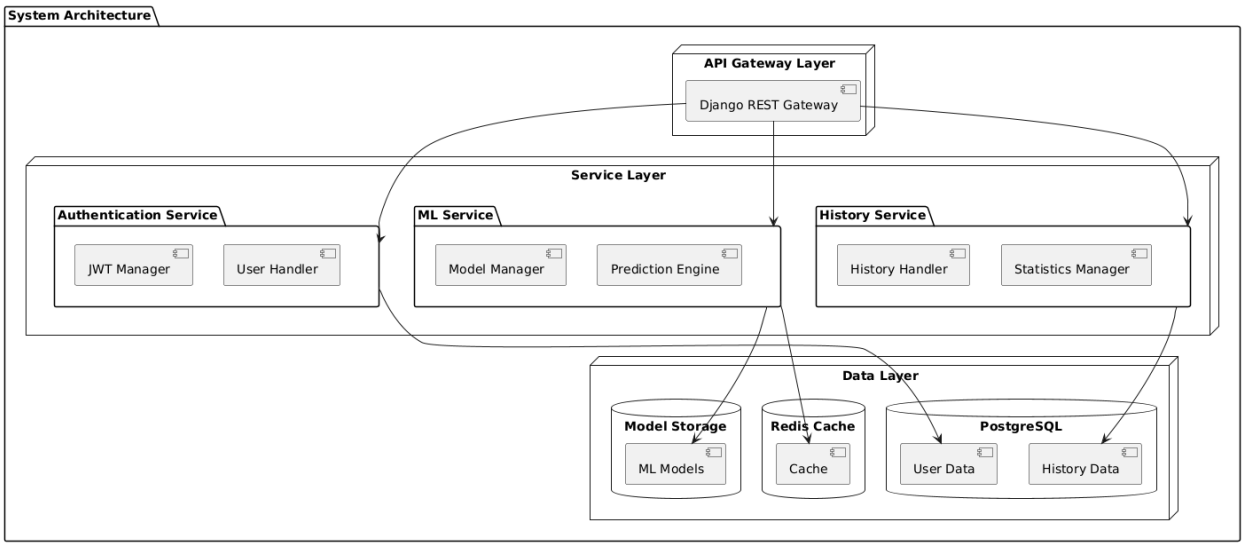


Рисунок 3.6 – Діаграма мікросервісної архітектури системи

Authentication Service відповідає за реєстрацію, автентифікацію та авторизацію користувачів. Цей сервіс реалізує JWT-based механізм безпеки, що дозволяє ефективно управляти сесіями користувачів та забезпечувати безпечний

доступ до API. При зростанні кількості користувачів саме цей сервіс може бути масштабований горизонтально для обробки збільшеного потоку запитів на авторизацію.

ML Service є ключовим компонентом системи, який виконує ресурсомісткі операції з аналізу текстів та класифікації новин. Цей сервіс обробляє запити паралельно та може бути масштабований як горизонтально (збільшення кількості екземплярів), так і вертикально (збільшення обчислювальної потужності, особливо GPU ресурсів для роботи з нейромережевими моделями). Структура ML сервісу була представлена в вигляді діаграми на рисунку 3.7.

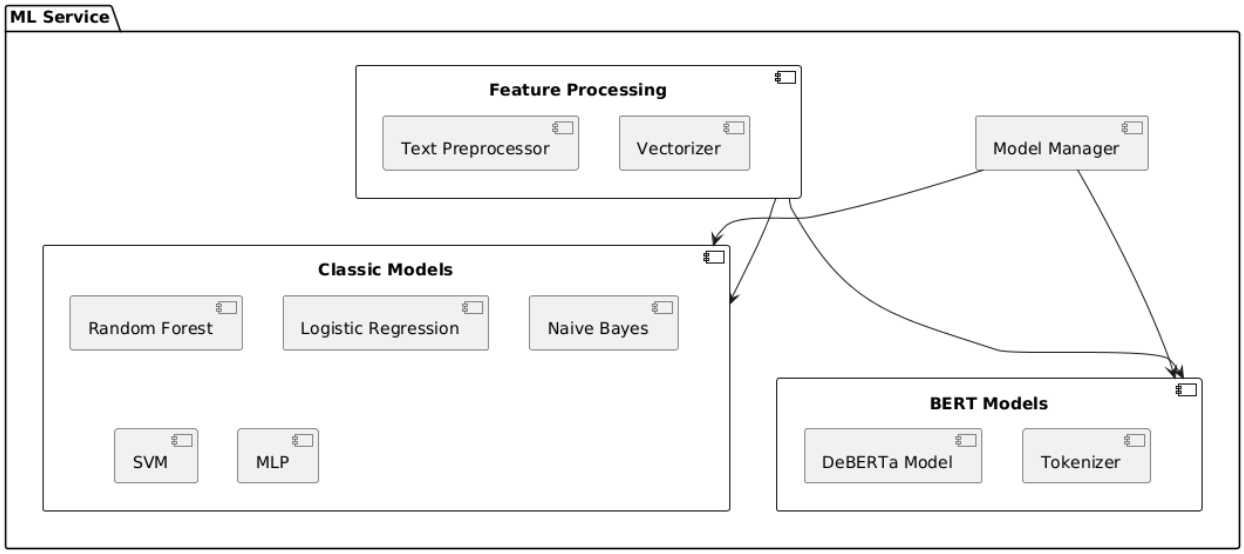


Рисунок 3.7 – Діаграма структури ML сервісу

History Service забезпечує функціонал для збереження та аналізу історії перевірок. Цей сервіс оптимізований для роботи з базою даних та може бути масштабований відповідно до росту об'єму даних.

3.2 Вибір та обґрунтування моделей машинного навчання

Для розв'язання комплексної проблематики верифікації інформаційного контенту було імплементовано дуальний підхід до моделювання: традиційні алгоритми статистичного навчання та передові нейроархітектури на базі багатошарових атентивних механізмів. Кожен із цих підходів має свої особливості

архітектури та принципи роботи, що впливає на їх здатність аналізувати текстові дані[17].

3.2.1 Порівняльний аналіз класичних моделей

У контексті класичних моделей машинного навчання розглянуто п'ять основних архітектур, а саме: баєсівський імовірнісний класифікатор, ансамблевий метод випадкового лісу, багатошаровий персептрон з повнозв'язними шарами, логістична регресія з градієнтним спуском та метод опорних векторів з оптимізацією гіперплощини[18].

Random Forest представляє собою ансамблеву модель, яка складається з множини деревоподібних класифікаторів. Кожне дерево будується на основі стохастично згенерованої підмножини навчального корпусу та атрибутів, що забезпечує різноманітність прогнозів та зменшує ризик перенавчання. Архітектура Random Forest дозволяє паралельно обробляти дані та автоматично визначати важливість ознак.

Логістична регресія використовує лінійну архітектуру з сигмоїдною функцією активації. Модель трансформує вхідні ознаки в логіти, які потім перетворюються в ймовірності належності до класів. Простота архітектури забезпечує високу інтерпретованість моделі та можливість аналізу впливу кожної ознаки на кінцеве рішення.

Naive Bayes базується на теоремі Баєса та припущенні про незалежність ознак. Архітектура моделі передбачає обчислення умовних ймовірностей для кожної ознаки відносно класів, що робить її особливо ефективною для обробки текстових даних, де важлива частота появи слів.

Multi-layer Perceptron представляє собою нейронну мережу прямого поширення з кількома прихованими шарами. Архітектура включає вхідний шар, що відповідає розмірності векторизованого тексту, приховані шари з нелінійними функціями активації та вихідний шар з двома нейронами для бінарної класифікації.

Support Vector Machine використовує архітектуру, яка трансформує вхідні дані у високорозмірний простір ознак за допомогою ядерних функцій. Це дозволяє

знаходити оптимальну гіперплощину для розділення класів, навіть якщо дані нелінійно роздільні у початковому просторі.

Як показано в Таблиці 3.1, архітектурні особливості класичних моделей узагальнено за основними параметрами, що дозволяє порівняти їх ключові характеристики та вибрати найбільш підходящу модель для конкретної задачі[19].

Таблиця 3.1 – Архітектурні особливості класичних моделей

Модель	Тип архітектури	Основні компоненти	Особливості обробки даних	Масштабованість
RF	Ансамблева	Дерева рішень	Паралельна обробка	Висока
LR	Лінійна	Сигмоїдна функція	Послідовна обробка	Середня
NB	Імовірнісна	Баєсові множники	Незалежна обробка ознак	Висока
MLP	Багатошарова	Нейронні шари	Послідовна обробка з прямим та зворотним поширенням	Середня
SVM	Кернельна	Ядерні функції	Трансформація простору ознак	Низька

Крім архітектурних особливостей, важливо також враховувати практичні аспекти використання моделей, такі як можливість розпаралелювання, вимоги до попередньої обробки даних та гнучкість архітектури. У Таблиці 3.2 наведено характеристики моделей з цих точок зору.

Таблиця 3.2 – Характеристики архітектури моделей

Модель	Можливість розпаралелювання	Вимоги до попередньої обробки	Гнучкість архітектури
RF	Висока	Середні	Висока
LR	Низька	Низькі	Низька
NB	Висока	Низькі	Низька
MLP	Середня	Високі	Висока
SVM	Низька	Високі	Середня

З точки зору архітектури та моделювання, кожен підхід має свої переваги та обмеження. Random Forest забезпечує найбільшу гнучкість та масштабованість, дозволяючи ефективно обробляти великі обсяги даних. Logistic Regression та Naive Bayes мають просту архітектуру, що робить їх легкими для впровадження та підтримки. MLP надає можливість моделювати складні нелінійні залежності, але потребує ретельного налаштування архітектури. SVM забезпечує потужні можливості для роботи з нелінійними даними, але має обмеження щодо масштабованості на великих наборах даних.

3.2.2 Порівняння архітектур на основі трансформерів

У галузі обробки природної мови (NLP) трансформерні архітектури стали визначним проривом, здійснивши революцію в способі обробки текстових даних[20]. На відміну від попередніх архітектур нейронних обчислень, таких як рекурентні часові моделі (RNN) та згорткові просторові мережі (CNN), трансформери реалізують багатовимірний механізм самоатентивності, який дозволяє моделі одночасно аналізувати всі елементи вхідної послідовності та визначати їх взаємозв'язки.

Основними представниками цього класу моделей є Bidirectional Encoder Representations from Transformers, що встановив новий стандарт у обробці природної мови, RoBERTa (Robustly Optimized BERT Approach), що покращила процес навчання BERT, XLNet, який запровадив автореґресивне попереднє навчання, T5 (Text-to-Text Transfer Transformer), що уніфікував різні NLP задачі в єдиний формат, та DeBERTa, яка представила інноваційний підхід до механізму уваги[21]. То ж було обрано модель DeBERTa (Decoding-enhanced BERT with disentangled attention), тому що вона демонструє революційний прорив в еволюції трансформерної топології нейронних обчислень. Ця модель була розроблена з метою подолання обмежень попередніх архітектур та підвищення ефективності обробки текстової інформації[22].

Роз'єднаний механізм уваги (Disentangled Attention) пропонує принципово новий підхід до обробки контенту та позиційної інформації. Він передбачає розділення векторних представлень на семантичні та позиційні компоненти, що

покращує здатність моделі розуміти складні лінгвістичні конструкції. Це дозволяє більш точно визначати взаємозв'язки між віддаленими елементами тексту та зменшує вплив позиційних артефактів на якість обробки.

Удосконалене позиційне кодування включає впровадження двонаправленого відносного позиційного кодування, що сприяє кращому розумінню ієрархічної структури тексту. Це підвищує ефективність обробки документів різної довжини та покращує здатність моделі захоплювати далекі залежності. В результаті забезпечується більш стабільна робота з довгими послідовностями.

Розширений механізм маскування передбачає вдосконалену стратегію маскування для попереднього навчання, використовуючи градієнтне заміщення для покращення процесу навчання. Це веде до більш ефективного передбачення маскованих токенів і покращує розуміння контекстуальних зв'язків. Додатково підвищується стійкість моделі до шумів та спотворень у вхідних даних.

Архітектурні оптимізації спрямовані на ефективніше використання обчислювальних ресурсів та покращену збіжність при навчанні. Оптимізована структура нейронної мережі та вдосконалений механізм регуляризації забезпечують кращу масштабованість на великих наборах даних.

Архітектурні особливості різних моделей обробки природної мови, таких як BERT, RoBERTa, XLNet, T5 і DeBERTa були представлені в таблиці 3.3.

Таблиця 3.3 – Порівняння архітектурних особливостей трансформерних моделей

Архітектурна особливість	BERT	RoBERTa	XLNet	T5	DeBERTa
Тип механізму уваги	Стандартний	Стандартний	Авторегресивний	Енкодер-декодер	Роз'єднаний
Позиційне кодування	Абсолютне	Абсолютне	Відносне	Абсолютне	Відносне з покращенням
Стратегія маскування	Базова	Динамічна	Пермістивна	Прогресивна	Розширена
Розмір контексту	512 токенів	512 токенів	512 токенів	512 токенів	2048 токенів

Кінець таблиці 3.3

Обробка довгих залежностей	Базова	Покращена	Хороша	Хороша	Відмінна
Семантичний аналіз	Базовий	Покращений	Покращений	Хороший	Розширений
Синтаксичний аналіз	Базовий	Базовий	Хороший	Базовий	Відмінний
Масштабованість	Середня	Висока	Висока	Висока	Надвисока

По основним характеристикам, що включають в себе тип механізму уваги, позиційне кодування, стратегію маскування, розмір контексту, обробку довгих залежностей, семантичний та синтаксичний аналіз, а також масштабованість, можна побачити, що DeBERTa вирізняється розширеною обробкою довгих залежностей, високою точністю синтаксичного аналізу та надвисокою масштабованістю, що робить її ефективною для роботи з великими текстовими даними.

Також було представлено результати тестування моделей на основі їхніх архітектурних характеристик у таблиці 3.4. Це тестування охоплює базові та покращені можливості, що стосуються семантичного аналізу, синтаксичного розбору та обробки складних текстових залежностей. Наприклад, модель T5 демонструє хороші результати в семантичному аналізі та прогресивну стратегію маскування, що робить її придатною для завдань перекладу й текстового узагальнення.

Таблиця 3.4 – Порівняння ключових характеристик архітектур

Характеристика	BERT	RoBERTa	XLNet	T5	DeBERTa
Біспрямованість	Так	Так	Частково	Так	Так
Попереднє навчання	MLM	MLM	PLM	T2T	Розширений MLM
Розмір моделі	Середній	Великий	Великий	Дуже великий	Оптимізований
Швидкість інференсу	Середня	Середня	Низька	Низька	Висока

Кінець таблиці 3.4

Ефективність навчання	Базова	Висока	Середня	Висока	Надвисока
Гнучкість архітектури	Середня	Середня	Висока	Висока	Надвисока

DeBERTa-v3 представляє собою подальший розвиток базової архітектури, впроваджуючи оптимізований механізм роз'єднаної уваги, вдосконалене позиційне кодування з покращеною обробкою контексту, більш ефективну стратегію попереднього навчання, покращену обробку рідкісних слів та фраз, а також розширені можливості узагальнення на нових даних.

Одним із ключових досягнень DeBERTa-v3 є оптимізований механізм роз'єднаної уваги. Цей підхід дозволяє моделі більш точно захоплювати семантичні та позиційні залежності між словами в тексті. На відміну від попередніх моделей, які часто об'єднували контентну та позиційну інформацію, DeBERTa-v3 обробляє їх окремо. Це призводить до глибшого розуміння складних лінгвістичних конструкцій та покращує здатність моделі розуміти нюанси мови. Така особливість є особливо корисною для задач, де важливим є тонке розрізнення змісту, як-от виявлення фейкових новин.

Крім того, вдосконалене позиційне кодування з покращеною обробкою контексту дозволяє моделі ефективніше розуміти відносини між віддаленими словами в реченні. Це досягається шляхом покращення способу, яким позиційна інформація кодується та використовується. В результаті модель краще захоплює ієрархічну структуру мови та більш ефективно обробляє далекі залежності. Це особливо важливо при роботі з довгими текстами, такими як новинні статті, де взаємодія між різними частинами тексту може бути критичною для розуміння загального змісту.

Більш ефективна стратегія попереднього навчання в DeBERTa-v3 сприяє швидшій конвергенції моделі та підвищенню її загальної точності. Використовуючи вдосконалені методи маскування та оптимізації, модель стає більш здатною до передбачення маскованих токенів та розуміння контекстуальних

зв'язків. Це також покращує обробку рідкісних слів та фраз, що є критичним для задач, де важливими є деталі та специфічна термінологія.

Розширені можливості узагальнення на нових даних роблять DeBERTa-v3 особливо цінною для виявлення фейкових новин. Модель здатна адаптуватися до нових патернів та структур, які можуть з'являтися в текстах, що дозволяє ефективно розпізнавати дезінформацію навіть у незнайомих контекстах. Це забезпечує актуальність моделі в динамічному інформаційному середовищі, де методи поширення фейкових новин постійно еволюціонують.

Для задачі виявлення фейкових новин архітектура DeBERTa-v3 надає кілька ключових переваг[23]. По-перше, поглиблений семантичний аналіз моделі веде до кращого розуміння контексту та нюансів мови. Модель ефективно виявляє суперечності в тексті та обробляє складні лінгвістичні конструкції, які часто зустрічаються у фейкових новинах. Це дозволяє більш точно ідентифікувати приховані ознаки дезінформації та маніпуляції, підвищуючи надійність результатів аналізу.

По-друге, DeBERTa-v3 демонструє високу робастність та надійність. Модель стійка до шумів у вхідних даних та стабільно працює з текстами різної довжини, включаючи неформатований або спотворений текст. Це особливо важливо при аналізі інформації з різних джерел, де якість та формат даних можуть суттєво відрізнятися. Стійкість до таких варіацій забезпечує надійність моделі в реальних умовах застосування.

По-третє, масштабованість та ефективність DeBERTa-v3 дозволяють оптимально використовувати обчислювальні ресурси. Модель забезпечує швидку обробку великих обсягів даних та підтримує можливість паралельної обробки багатьох запитів. Це критично для систем, які повинні оперативно реагувати на потік новин та швидко ідентифікувати фейкову інформацію. Висока продуктивність моделі робить її придатною для інтеграції в масштабні інформаційні платформи.

Нарешті, гнучкість та адаптивність DeBERTa-v3 дозволяють тонко налаштовувати модель під специфіку конкретної задачі. Модель ефективно працює

з різними типами текстових даних та здатна адаптуватися до нових патернів фейкових новин. Це особливо важливо в сучасному інформаційному середовищі, де методи дезінформації постійно змінюються. Здатність моделі до адаптації забезпечує її довгострокову ефективність та актуальність.

Порівняно з попередніми моделями, такими як BERT та RoBERTa, DeBERTa-v3 демонструє вищу точність та робастність у задачах обробки природної мови. Вона перевершує їх у здатності розуміти складні лінгвістичні структури та виявляти приховані смислові зв'язки. Крім того, DeBERTa-v3 відрізняється вищою швидкістю інференсу та кращою масштабованістю, що робить її більш придатною для практичного застосування в умовах обмежених ресурсів та необхідності швидкої обробки даних.

Наприклад, у контексті виявлення фейкових новин, DeBERTa-v3 здатна більш ефективно аналізувати тексти, які містять складні метафори, іронію або сарказм, що часто використовується для маскування дезінформації. Модель може виявляти невідповідності та суперечності в інформації, які можуть бути ознаками неправдивих повідомлень. Це досягається завдяки глибокому розумінню семантичних та синтаксичних аспектів мови.

Крім того, DeBERTa-v3 демонструє високу стійкість до різних форм текстових маніпуляцій, таких як навмисні орфографічні помилки, використання синонімів або складних граматичних конструкцій для обходу автоматичних систем перевірки. Модель здатна розпізнавати сутність повідомлення навіть у присутності таких спотворень, що підвищує її ефективність у реальних умовах застосування.

Узагальнюючи, DeBERTa-v3 демонструє квантовий стрибок у розвитку нейроархітектурної топології глибинного навчання, пропонуючи унікальні особливості, які безпосередньо вирішують проблеми, властиві задачі детекції дезінформаційного контенту. Її вдосконалені механізми самоатентивності, інноваційне позиційне ембедування, високоефективні стратегії тренування та адаптивна пластичність роблять її провідною моделлю для задач, що вимагають глибокого розуміння мови та стійкої роботи в умовах різноманітних даних.

Таким чином, вибір DeBERTa-v3 для вирішення задачі детекції дезінформаційного контенту є обґрунтованим на основі її технічних переваг та практичної ефективності. Модель забезпечує високу точність, швидкість та надійність, що є критичним для успішного протистояння поширенню дезінформації та підтримки достовірності інформаційного простору.

3.2.3 Оптимізація трансформерної моделі DeBERTa-v3

У процесі інженерної імплементації системи детекції дезінформаційного контенту ключовий фокус було зосереджено на оптимізації гіперпараметрів моделі DeBERTa-v3[24]. Це налаштування є критично важливим етапом, оскільки правильний вибір гіперпараметрів безпосередньо впливає на продуктивність моделі, її точність та здатність до узагальнення на нових даних.

Швидкість навчання (learning rate) була встановлена на рівні 2e-5. Це значення було обрано оскільки воно забезпечує оптимальний баланс між швидкістю збіжності моделі та стабільністю процесу навчання. Занадто висока швидкість навчання може призвести до коливань та нестабільності, тоді як занадто низька уповільнює процес навчання і може застрягнути в локальних мінімумах.

Розмір батчу (batch size) було обрано рівним 8, що дозволяє оптимально використовувати доступну пам'ять GPU та забезпечує ефективну обробку даних. Менший розмір батчу сприяє більш стабільним оновленням вагів та кращій узагальнювальній здатності моделі, особливо при обмежених ресурсах.

Кількість епох навчання була обмежена двома епохами з можливістю ранньої зупинки. Це рішення було прийнято на основі аналізу метрик продуктивності на валідаційному наборі даних. Якщо модель починає перенавчатися або покращення метрик сповільнюється, навчання зупиняється раніше. Такий підхід запобігає перенавчанню та сприяє кращій генералізації моделі на нових даних.

Для оптимізації вагів моделі використовувався оптимізатор AdamW з параметром weight_decay=0.01. Введення вагового спаду (weight decay) сприяє регуляризації моделі, зменшуючи перевагу великих вагів і тим самим запобігаючи перенавчанню. Оптимізатор AdamW комбінує переваги адаптивних методів навчання з можливістю ефективної регуляризації.

Стратегія прогріву (warmup) передбачала використання 10% початкових ітерацій для поступового збільшення швидкості навчання від нуля до встановленого значення. Це допомагає моделі стабільно починати процес навчання, уникаючи різких змін градієнтів, які можуть негативно вплинути на збіжність. Такий поступовий підхід забезпечує більш гладке оновлення вагів на початкових етапах навчання.

Ретельне налаштування цих гіперпараметрів було здійснено шляхом експериментів та аналізу результатів на тренувальних та валідаційних наборах даних. Було враховано специфіку задачі детекції дезінформаційного контенту, а саме: необхідність обробки великих текстових масивів, складність лінгвістичних конструкцій та різноманітність джерел даних.

У результаті такого підходу модель DeBERTa-v3 продемонструвала високі показники точності та надійності. Оптимізовані гіперпараметри сприяли покращенню продуктивності моделі без надмірного споживання обчислювальних ресурсів. Це підтверджує важливість та ефективність процесу fine-tuning гіперпараметрів у контексті розробки сучасних моделей глибинного навчання.

Важливо зазначити, що процес налаштування гіперпараметрів є ітеративним і може вимагати додаткових коригувань при зміні умов або даних. Тому модель була спроєктована з можливістю гнучкого налаштування, що дозволяє адаптувати її до різних сценаріїв застосування та забезпечувати стабільну роботу в динамічному інформаційному середовищі.

Таким чином, проведене налаштування гіперпараметрів є невід'ємною частиною архітектури та моделювання інтелектуальної системи верифікації інформаційного контенту. Воно гарантує максимальну точність нейромережевих обчислень та підвищує потенціал системи у виявленні дезінформаційних патернів у різноманітних контекстуальних сценаріях.

3.3 Вибір технологій та інструментів розробки

3.3.1 Серверна частина (Backend)

Високорівнева мова програмування Python була імплементована як фундаментальна технологія для розробки серверної архітектури з урахуванням її масштабованої екосистеми фреймворків нейромережевого моделювання та інструментарію обробки великих даних. Python забезпечує комплексний інструментарій лінгвістичного процесингу, містить вичерпну колекцію фреймворків нейромережевого моделювання та вирізняється елегантністю синтаксичних конструкцій, що оптимізує процес розробки.

Django обрано як вебфреймворк для створення серверної частини платформи. Цей вибір обумовлений його надійністю, масштабованістю та вбудованими засобами безпеки. Фреймворк Django забезпечує комплексну екосистему інструментарію для розробки високонавантажених веб-платформ, інтегруючи багаторівневу систему контролю доступу, інтерфейс адміністрування та потужний ORM-шар для абстрагування операцій з persisted data.

Інфраструктурний компонент Django REST Framework імплементується для розгортання RESTful архітектури, що реалізує безшовну інтеграцію клієнт-серверної взаємодії. Технологічний стек забезпечує високопродуктивні механізми трансформації даних, верифікації вхідних потоків та автоматизованої специфікації API-ендпоїнтів, що радикально оптимізує життєвий цикл розробки та підтримки програмних інтерфейсів.

Python бібліотеки для машинного навчання:

– pandas забезпечує ефективну обробку та аналіз структурованих даних, бібліотека надає зручні інструменти для маніпуляції даними, їх очищення та підготовки до машинного навчання, що особливо важливо при роботі з текстовими даними новин[25];

– scikit-learn застосовується для імплементації традиційних архітектур статистичного навчання та препроцесингу датасетів, фреймворк забезпечує уніфіковану парадигму взаємодії з різноманітними методами предиктивної

аналітики, включаючи ансамблевий метод випадкового лісу, логістичну регресію з градієнтним спуском, метод опорних векторів та інші;

– torch та transformers є фундаментальними компонентами екосистеми для імплементації архітектури DeBERTa, torch реалізує масштабовану інфраструктуру нейромережевих обчислень з апаратною акселерацією на GPU, а transformers пропонує екосистему претренованих нейроархітектур, інструментарій для їх тонкого налаштування та оптимізації гіперпараметрів, а також високорівневий програмний інтерфейс для промислового розгортання моделей, що робить її ключовою технологією для оперування багатошаровими атентивними нейромережами[26];

– NLTK (Natural Language Toolkit) імплементовано для фундаментальних операцій лінгвістичного процесингу, включаючи сегментацію текстових структур, елімінацію семантично нейтральних лексем та інші алгоритми препроцесингу текстового корпусу;

– imbalanced-learn інтегровано для нівелювання статистичної асиметрії розподілу класів у тренувальному датасеті, фреймворк забезпечує імплементацію алгоритму синтетичної мінорітарної вибірки SMOTE та комплексу методів гармонізації навчальних даних.

Інфраструктурні компоненти нейромережевої екосистеми:

– tensorboard використовується для візуалізації та моніторингу процесу навчання моделей, інструмент дозволяє відстежувати динаміку метрик, аналізувати розподіли параметрів та оптимізувати гіперпараметри моделей.

– CUDA забезпечує прискорення обчислень на GPU, що критично важливо для ефективного навчання та використання глибоких нейронних мереж, підтримка CUDA дозволяє значно пришвидшити процес навчання моделі DeBERTa та обробку великих обсягів текстових даних.

Обраний стек технологій для серверної частини забезпечує оптимальний баланс між продуктивністю, масштабованістю та зручністю розробки. Інтеграція Django з потужними бібліотеками машинного навчання дозволяє створити надійну та ефективну систему для виявлення фейкових новин.

3.3.2 Клієнтська частина (Frontend)

Для розробки клієнтської частини системи виявлення фейкових новин було обрано TypeScript як базовий інструментарій програмної інженерії. TypeScript розширює можливості JavaScript, впроваджуючи строгу типізацію, що радикально підвищує надійність кодової бази та оптимізує її підтримку. Використання TypeScript дозволяє ідентифікувати потенційні аномалії на етапі компіляції, забезпечує вичерпну самодокументованість програмних компонентів та посилює ефективність архітектурної реорганізації. Особливо критичним є те, що TypeScript реалізує передову інтеграцію з високонавантаженими фронтенд-фреймворками та екосистемними бібліотеками.

React було обрано як основний фреймворк для побудови користувацького інтерфейсу. Цей вибір обумовлений його компонентним підходом, що забезпечує створення масштабованої архітектури та реюзабельних інтерфейсних абстракцій. React забезпечує ефективне оновлення DOM завдяки використанню віртуального DOM, що оптимізує продуктивність системи. Декларативний підхід React до побудови інтерфейсу робить код більш передбачуваним та легшим для розуміння, а велика екосистема компонентів та інструментів пришвидшує процес розробки.

Для оркестрації глобального стейт-менеджменту імплементовано Redux – фреймворк, що реалізує предиктивний контейнер стану. Redux забезпечує централізоване персистентне сховище даних та уніфікований потік трансформацій, що є критичним для масштабованих аплікацій з комплексною архітектурою взаємодій. Для спрощення роботи з Redux було впроваджено Redux Toolkit (RTK), який надає набір інструментів для зменшення шаблонного коду та стандартизації архітектури. RTK включає вбудовані утиліти для налаштування сховища, створення редюсерів та обробки асинхронних дій, що значно пришвидшує розробку.

В частині стилізації та побудови користувацького інтерфейсу було обрано SCSS модулі. Цей підхід дозволяє створювати модульні та ізольовані стилі для компонентів, уникаючи конфліктів імен класів та забезпечуючи кращу організацію CSS-коду. SCSS розширює можливості звичайного CSS, додаючи підтримку

змінних, міксинів, вкладених правил та інших корисних функцій, що робить стилізацію більш гнучкою та підтримуваною. Модульний підхід до CSS також сприяє повторному використанню стилів та полегшує підтримку великих вебплатформ.

Обраний стек технологій для клієнтської частини формує сучасну та потужну основу для створення інтерактивного вебплатформи. Комбінація TypeScript, React, Redux та SCSS модулів забезпечує високу продуктивність, масштабованість та зручність розробки, дозволяючи створити якісний користувацький інтерфейс для системи виявлення фейкових новин. Використання цих технологій також гарантує хорошу підтримку з боку спільноти розробників та постійний розвиток інструментарію.

Висновки до розділу 3

Інтеграція та імплементація архітектурних рішень системи верифікації інформаційного контенту була реалізована на базі клієнт-серверної парадигми з використанням мікросервісної топології, що забезпечило оптимальну декомпозицію функціональних компонентів та ефективне масштабування обчислювальних ресурсів. Архітектурна модель передбачає розподілену інфраструктуру з чітко визначеними інтерфейсами взаємодії, що гарантує високу когезію компонентів при збереженні їх незалежності.

У контексті методологій машинного навчання було імплементовано дуальний підхід, що комбінує традиційні алгоритми статистичного навчання та передові нейроархітектури на базі багатошарових атентивних механізмів. Ключовим компонентом системи стала модель DeBERTa-v3, яка демонструє революційний прорив у розвитку трансформерної топології нейронних обчислень, забезпечуючи вдосконалені механізми самоатентивності та інноваційне позиційне ембедування для оптимальної обробки лінгвістичних патернів.

Технологічний стек розробки був сформований з урахуванням сучасних вимог до масштабованості та продуктивності систем. Серверна частина базується на високорівневій мові програмування Python з інтеграцією потужних фреймворків

нейромережевого моделювання, в той час як клієнтський компонент реалізовано з використанням TypeScript та React, що забезпечує створення інтуїтивного користувацького інтерфейсу з високою реактивністю та надійністю. Імплементована архітектура системи верифікації інформаційного контенту демонструє оптимальний баланс між технологічною інноваційністю та практичною ефективністю, створюючи надійну платформу для протидії поширенню дезінформації в сучасному цифровому просторі.

4 РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Реалізація та навчання моделей машинного навчання

4.1.1 Підготовка та попередня обробка даних

При розробці системи виявлення фейкових новин критично важливим етапом є правильна підготовка та обробка даних. Для тренування моделей було використано WELFake Dataset - спеціалізований набір даних, що містить розмічені новинні статті. Спочатку було виконано завантаження датасету:

```
# Завантаження датасету
data = pd.read_csv('WELFake_Dataset.csv', usecols=['title', 'text', 'label'])
```

Для оптимізації використання пам'яті одразу обрано лише необхідні стовпці: заголовок новини (title), текст новини (text) та мітку класу (label).

Після завантаження проведено детальний аналіз якості даних:

```
# Перевірка пропущених значень
missing_both = data[data['title'].isnull() & data['text'].isnull()]
print("Рядки з пропусками в двох колонках відразу 'title' и 'text':")
print(missing_both)
# Заповнення можливих пропусків порожніми рядками
data['title'] = data['title'].fillna('')
data['text'] = data['text'].fillna('')
```

Аналіз показав загальну кількість записів, що дорівнює 72134, та високу якість вхідних даних, а саме відсутні записи з пропущеними значеннями в обох полях одночасно. Для забезпечення стабільності подальшої обробки всі потенційні пропуски заповнено порожніми рядками.

Важливим аспектом аналізу стала перевірка розподілу класів. Виявлено незначний дисбаланс:

– 37106 фейкових новин (мітка 1);
– 35028 правдивих новин (мітка 0).

Додатково на рисунку 4.1 представлено графік у вигляді «ящиків з вусами», який демонструє розподіл кількості слів у текстах залежно від їхнього класу

— спам або не спам, цей графік був створений для візуалізації особливостей текстових даних, зокрема для виявлення відмінностей між класами за кількістю слів.

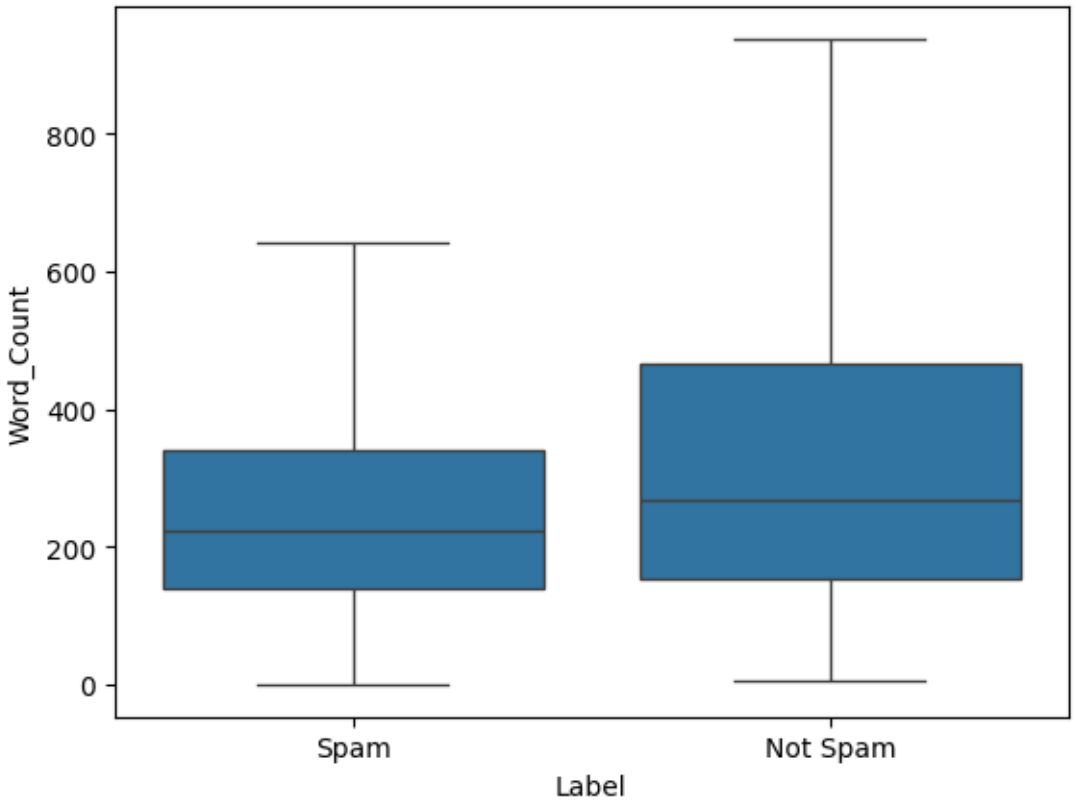


Рисунок 4.1 – Графік розподілу кількості слів у текстах залежно від їхнього класу

Графік показує, що різниця між цими двома класами є невеликою, хоча можна помітити тенденцію до більшої варіативності в текстах класу «Not Spam».

Для якісної попередньої обробки текстових даних реалізовано спеціалізовану функцію:

```
def preprocess_text(text):
    if isinstance(text, str):
        # Переведення в нижній регістр
        text = text.lower()
        # Видалення URL
        text = re.sub(r'http\S+|www\S+|https\S+', '', text,
flags=re.MULTILINE)
        # Видалення email адрес
        text = re.sub(r'\S+@\S+', '', text)
        # Видалення спеціальних символів, збереження заперечень
```

```
            text = re.sub(r'[^a-zA-Z\s\'not]', ' ', text)
            # Заміна множинних пробілів
            text = ' '.join(text.split())
            return text
        return ' '
```

Функція preprocess_text розроблена для стандартизації та очищення текстових даних, що особливо важливо в задачах обробки природної мови (NLP) і машинного навчання. Головною цілю для цієї функції служить підготування тексту до подальшого аналізу або використання в моделях машинного навчання, усуваючи небажані елементи, які можуть викликати шум або спотворити результати. В цій функції було зроблено ряд завдань, а саме:

– зведення всіх символів до одного формату, що полегшило порівняння слів і зменшило розмір словника.

– видалення посилань (URL) зменшило шум у даних і спростило текст, тому що часто посилання не несуть корисної інформації для текстового аналізу, але можуть займати значну частину тексту.

– видалення email-адреси, дозволило зосередитися на смислових елементах тексту, тому що вони також є унікальними елементами, які рідко потрібні для аналізу тексту;

– видалення множинних пробілів, цей крок перетворив будь-яку кількість пробілів у єдиний роздільник між словами, що полегшив обробку.

В рисунку 4.1 наведено інформацію про успішне підготування тексту.

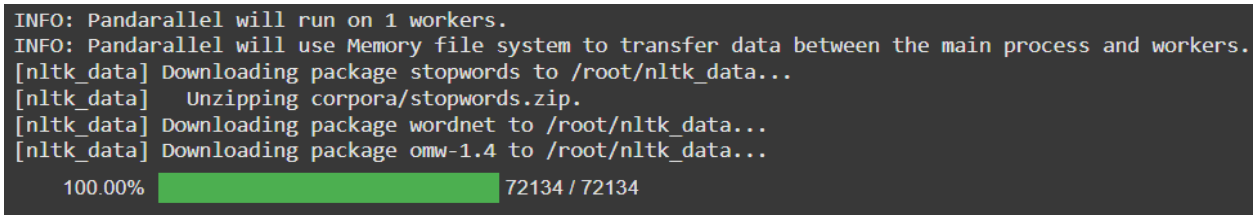


Рисунок 4.2 – Логування та моніторинг процесу підготування тексту

Також одним із важливих аспектів реалізації стало врахування специфіки роботи з текстовими даними, зокрема, збереження слів заперечення. Це було

зроблено для забезпечення точності та коректності аналізу тексту, особливо у контексті задач, пов'язаних із визначенням тональності та правдивості новин.

Саме тому під час підготовки списку стоп-слів, який використовується для очищення тексту, було розроблено механізм, що виключає слова заперечення зі списку стоп-слів, що забезпечує їх збереження у тексті для подальшого аналізу:

```
# Отримання стоп-слів із видаленням заперечень
stop_words = set(stopwords.words('english'))
negation_words = {'no', 'not', 'nor', 'neither', "n't", 'never', 'none'}
stop_words = [word for word in stop_words if word not in negation_words]
# Об'єднання заголовка і тексту
df['title_text'] = df['processed_title'] + ' ' + df['processed_text']
```

На рисунку 4.2 представлено дві хмарки слів, які демонструють найчастіше вживані терміни у текстах для двох класів: фейкові та реальні новини, ця візуалізація була створена для аналізу лексичного складу текстів кожного класу та виявлення ключових слів, які можуть допомогти у диференціації цих класів.

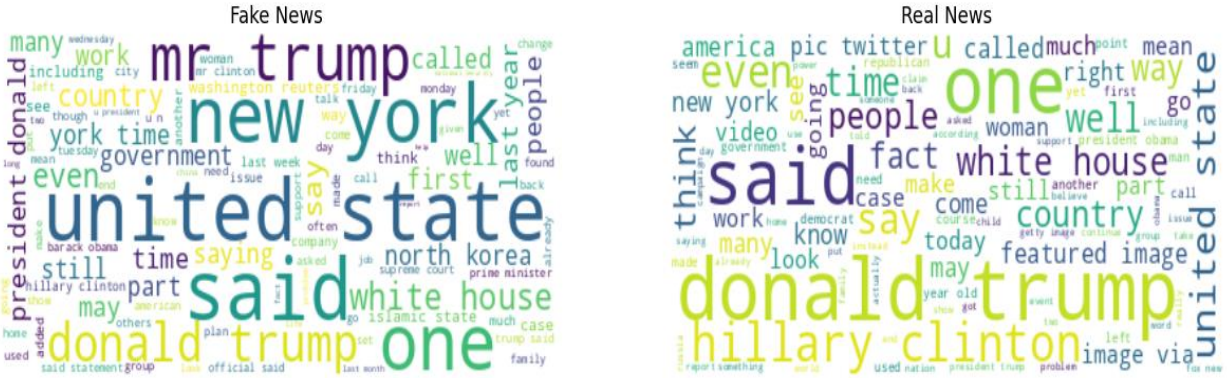


Рисунок 4.3 – Хмари найчастіше вживаних термінів у текстах новин

У хмарках слів для обох класів можна побачити значну кількість однакових ключових слів, таких як «donald trump», «united state», «said», що свідчить про те, що обидва класи новин часто охоплюють схожі теми, але у фейкових новинах частіше зустрічаються слова, пов'язані з емоційно забарвленими або маніпулятивними фразами, а у правдивих новинах більше уваги приділяється фактичній інформації.

Ця візуалізація допомогла краще зрозуміти структуру текстів і підтвердила доцільність використання TF-IDF або подібних підходів для виділення важливих слів і фраз, які сприяють визначенню класу новини.

Наступним критично важливим етапом стала трансформація лінгвістичних структур у математичні репрезентації. Для цього було обрано метод TF-IDF (Term Frequency-Inverse Document Frequency), який забезпечує конвертацію текстового корпусу в багатовимірний векторний простір, зберігаючи семантичну значущість лексичних компонентів у контекстуальному вимірі. Векторизація виконана з оптимізованими параметрами:

```
tfidf = TfidfVectorizer(
    stop_words=stop_words,
    max_features=10000,
    ngram_range=(1, 2),
    min_df=2,
    max_df=0.95
)
```

Параметри TF-IDF векторизатора:

– stop_words=stop_words – використано список стоп-слів, очищений від слів заперечення, щоб зберегти критично важливі терміни для аналізу;

– max_features=10000 – обмежено кількість унікальних термінів, які враховуються у векторизації, до 10 000, це дозволило зменшити розмірність вхідних даних і уникнути надлишкової інформації, яка може зашкодити моделі;

– ngram_range=(1, 2) - використано як уніграми (окремі слова), так і біграми (пари послідовних слів), щоб зберегти контекст тексту саме тому, враховувалося не тільки значення окремих термінів, але й їх взаємозв’язки в межах тексту;

– min_df=2 – термін вважається значущим лише тоді, коли він зустрічається хоча б у двох документах, це усунуло надто рідкісні слова, які можуть бути випадковими або унікальними, але не несуть корисної інформації для моделі;

– max_df=0.95 – було виключено терміни, які з’являються у більш ніж 95% документів, тому що такі слова зазвичай є дуже поширеними й не додають унікальної інформації.

Після векторизації текстові дані було векторизовано за допомогою TF-IDF, що дозволило перетворити їх у числовий формат, придатний для використання у моделях машинного навчання, забезпечуючи врахування ваги термінів залежно від їх частотності як у тексті, так і в усьому корпусі:

```
X = tfidf.fit_transform(data['Content'])
y = data['Label']
```

Для забезпечення якісного навчання моделей виконано розділення даних на тренувальну та тестову вибірки, то ж дані було розділено на навчальну вибірку (80%), яка використовується для тренування моделі та тестову вибірку (20%), яка використовується для оцінки її якості, а опція stratify=y забезпечує збереження пропорцій між класами у навчальному та тестовому наборах, що важливо для роботи з незбалансованими даними.

Оскільки вхідні дані містили невеликий дисбаланс між класами (один із класів мав менше зразків), для навчальної вибірки було застосовано метод SMOTE:

```
smote = SMOTE(random_state=42)
X_train_balanced,y_train_balanced= smote.fit_resample(X_train,y_train)
```

Така попередня обробка забезпечила якісну підготовку даних для подальшого навчання як класичних моделей машинного навчання, так і трансформерної моделі DeBERTa-v3.

4.1.2 Навчання класичних моделей машинного навчання

Реалізація системи виявлення фейкових новин включала навчання п'яти класичних моделей машинного навчання. Кожна модель була обрана з урахуванням її специфічних переваг для задачі класифікації текстів, а також перевірено, які дані дали більшу точність моделей. Для кожної моделі були обрані та налаштовані параметри з метою покращення їх продуктивності.

Першою була реалізована Random Forest Classifier:

```
RandomForestClassifier(
    n_estimators=200,
    class_weight='balanced',
    random_state=42
)
```

Пояснення аргументів:

– n_estimators=200 – збільшена кількість дерев для підвищення стабільності та зменшення варіації моделі;

– class_weight='balanced' – автоматичне зважування класів для врахування можливого дисбалансу в даних;

– random_state=42 – фіксація початкового стану генератора випадкових чисел для відтворюваності результатів.

Другою було реалізовано Logistic Regression:

```
LogisticRegression(class_weight='balanced',max_iter=1000,random_state
=42)
```

Пояснення аргументів:

– class_weight='balanced' – зважування класів для компенсації дисбалансу;

– max_iter=1000 – збільшення максимальної кількості ітерацій для забезпечення збіжності алгоритму;

– random_state=42 – для відтворюваності.

Наступною було реалізовано Multinomial Naive Bayes:

```
MultinomialNB(alpha=0.1)
```

Пояснення аргументів:

– alpha=0.1 – налаштування параметра згладжування Лапласа для зменшення впливу нульових ймовірностей.

Четвертою було реалізовано MLPClassifier:

```
MLPClassifier( hidden_layer_sizes=(100, 50), max_iter=1000,
random_state=42, early_stopping=True )
```

Пояснення аргументів:

– hidden_layer_sizes=(100, 50) – визначення архітектури нейронної мережі з двома прихованими шарами;

– max_iter=1000 – максимальна кількість ітерацій для навчання;

– early_stopping=True – зупинка навчання при відсутності покращення на валідаційному наборі, що запобігає перенавчанню;

– random_state=42 – для відтворюваності.

Останньою з класичних моделей було реалізовано Support Vector Machine:

```
SVC( kernel='linear', class_weight='balanced',
```

```
random_state=42, probability=True )
```

Пояснення аргументів:

– kernel='linear' – використання лінійного ядра, що підходить для розріджених високорозмірних даних після TF-IDF векторизації;

– class_weight='balanced' – врахування дисбалансу класів;

– probability=True – дозвіл обчислення ймовірностей для кожного класу;

– random_state=42 – для відтворюваності.

Для моделей, які не повертають достовірних ймовірностей, використовується CalibratedClassifierCV для калібрування вихідних ймовірностей, дозволяючи отримати більш точні ймовірності, що важливо для оцінки моделі та прийняття рішень:

```
if not isinstance(model, MultinomialNB):
    calibrated_model = CalibratedClassifierCV(model, cv=5)
    calibrated_model.fit(X_train, y_train)
else:
    calibrated_model = model
    calibrated_model.fit(X_train, y_train)
```

Використання крос-валідації з 5 фолдами (cv=5) для стабільності оцінок, а для MultinomialNB калібрування не потрібне, оскільки ця модель вже працює з ймовірностями.

Навчання моделей проведено на двох варіантах вхідних даних:

```
# Навчання на заголовках
print("Training models on titles only...")
for name, model in models.items():
    _ = train_improved_model(X_title, y, name, model, 'titles')
# Навчання на повних текстах
print("\nTraining models on combined features...")
for name, model in models.items():
    _ = train_improved_model(X_combined, y, name, model, 'combined')
```

Оцінки кожної моделі були виведено у вигляді тексту, для прикладу було відображено оцінку MLPClassifier на рисунку 4.4, але для більшої зручності було зведено всі оцінки моделей в єдину таблицю 4.1 та обрано тільки збалансовану точність для порівняння.

```
Training mlp on combined
Balanced accuracy: 0.9636

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.96      0.96      7377
           1       0.96      0.97      0.96      7466

    accuracy                           0.96     14843
   macro avg       0.96      0.96      0.96     14843
weighted avg       0.96      0.96      0.96     14843

Model saved as mlp_combined_improved.pkl
```

Рисунок 4.4 – Оцінка моделі MLPClassifier тренованого на заголовку та тексті новини

Таблиця 4.1 – Порівняння збалансованих оцінок класичних моделей машинного навчання

Модель	Збалансована точність моделі на основі комбінованого тексту	Збалансована точність моделі на основі тільки заголовків
RF	0,9648	0,9031
LR	0,9520	0,9025
NB	0,8658	0,8667
MLP	0,9636	0,9062
SVM	0,9633	0,9046

Результати показують, що використання повного тексту новин замість лише заголовків значно покращує точність класифікації для всіх моделей, крім Naive Bayes. Найкращі результати продемонстрував Random Forest на повних текстах з точністю 96.48%.

Після чого було збережено кожну модель та векторизатора для подальшого використання на серверній частині.

4.1.3 Налаштування та навчання моделі DeBERTa-v3

Для підвищення якості класифікації фейкових новин було реалізовано налаштування та створення моделі DeBERTa-v3. Вибір саме цієї архітектури обумовлений її високою ефективністю в задачах обробки природної мови та здатністю захоплювати складні семантичні зв'язки в тексті.

Використання GPU значно прискорює процес навчання глибоких нейронних мереж, тому було реалізовано перевірку на наявність та використання графічного модулю, а лістинг коду зображено на рис. 4.5.

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"\nUsing device: {device}")
if torch.cuda.is_available():
    print(f"GPU Name: {torch.cuda.get_device_name(0)}")
    print(f"Total GPU Memory: {torch.cuda.get_device_properties(0).total_memory / 1024**3:.2f} GB")
    print(f"Number of GPUs: {torch.cuda.device_count()}")
else:
    print("No GPU available, using CPU")
```

Рисунок 4.5 – Лістинг перевірочного коду щодо наявності та використання графічного модулю

Результати перевірки зображено на рисунку 4.6 та відповідно для подальшого навчання моделі буде використовуватися відеокарта NVIDIA Tesla T4 з 15 ГБ відеопам'яті.

```
Using device: cuda
GPU Name: Tesla T4
Total GPU Memory: 14.75 GB
Number of GPUs: 1
```

Рисунок 4.6 – Інформація про доступну для використання відеокарту

Далі дані були розділені на тренувальний, валідаційний та тестовий набори у співвідношенні 81% до 9% до 10% з використанням стратифікованого розбиття для збереження пропорцій класів. У процесі машинного навчання важливо правильно розподілити наявні дані, щоб модель могла навчитися ефективно і була об'єктивно оцінена. Розділення даних на тренувальний, валідаційний та тестовий набори є стандартною практикою, яка допомагає досягти цієї мети.

Для підготовки даних до моделі було створено клас FakeNewsDataset, який успадковує Dataset з torch і відповідає за токенізацію та підготовку батчів, лістинг коду для реалізації створення цього класу представлений на рис. 4.7.

Особливістю реалізації є використання спеціального токену «[SEP]» для розділення заголовка та тексту новини, що дозволяє моделі розрізняти ці дві важливі частини та враховувати їх окремо при аналізі, то ж модель отримує можливість аналізувати взаємозв'язок між заголовком та текстом, що є важливим для виявлення фейкових новин, де заголовки можуть бути маніпулятивними або вводити в оману.

По-друге, використання токенайзера моделі DeBERTa-v3 забезпечує правильну сегментацію та кодування тексту згідно з очікуваннями моделі. Це мінімізує ризик виникнення помилок під час навчання, пов'язаних з невідповідністю формату або розміру вхідних даних.

```python
class FakeNewsDataset(Dataset):
    def __init__(self, texts, titles, labels, tokenizer, max_length=512):
        self.texts = texts
        self.titles = titles
        self.labels = labels
        self.tokenizer = tokenizer
        self.max_length = max_length

    def __len__(self):
        return len(self.labels)

    def __getitem__(self, idx):
        title = str(self.titles[idx])
        text = str(self.texts[idx])

        # Combine title and text with special separator
        combined_text = title + " [SEP] " + text

        encoding = self.tokenizer(
            combined_text,
            truncation=True,
            padding='max_length',
            max_length=self.max_length,
            return_tensors='pt'
        )
        return {
            'input_ids': encoding['input_ids'].flatten(),
            'attention_mask': encoding['attention_mask'].flatten(),
            'labels': torch.tensor(self.labels[idx], dtype=torch.long)
        }
```

Рисунок 4.7 – Лістинг коду для створення класу FakeNewsDataset

Наступним важливим кроком є ініціалізація токенайзера та самої моделі з використанням попередньо навченої архітектури. Для цього було обрано модель microsoft/deberta-v3-base, яка відома своєю високою ефективністю в задачах

обробки природної мови та класифікації тексту, а лістинг коду для використання даної моделі продемонстровано на рисунку 4.8.

По-друге, ініціалізація моделі з параметром num_labels=2 автоматично налаштував вихідний шар моделі на дві категорії, що спростило процес навчання та знизило ризик помилок у конфігурації моделі, а це є особливо важливим для забезпечення коректного функціонування функції втрат та оптимізації.

```
model_name = "microsoft/deberta-v3-base"  # Using DeBERTa for better performance
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSequenceClassification.from_pretrained(
    model_name,
    num_labels=2,
    problem_type="single_label_classification"
)
```

Рисунок 4.8 – Ініціалізація моделі та токенайзера

По-третє, використання AutoTokenizer гарантувало, що текстові дані будуть токенізовані у спосіб, сумісний з очікуваннями моделі. Цей підхід знизив ймовірність виникнення помилок під час навчання, пов'язаних з некоректною токенізацією або невідповідністю словника моделі.

Про успішність ініціалізації моделі та токенайзеру було проінформовано та зображено на рисунку 4.9.

```
tokenizer_config.json: 100% 52.0/52.0 [00:00<00:00, 3.45kB/s]
config.json: 100% 579/579 [00:00<00:00, 38.5kB/s]
spm.model: 100% 2.46M/2.46M [00:00<00:00, 15.0MB/s]
pytorch_model.bin: 100% 371M/371M [00:01<00:00, 233MB/s]
```

Рисунок 4.9 – Завершення процесу ініціалізцації моделі та токенайзеру

У процесі тонкого налаштування моделі DeBERTa-v3 були встановлені параметри навчання за допомогою класу TrainingArguments. Ці параметри були ретельно обрані для забезпечення оптимальної продуктивності моделі та ефективного використання обчислювальних ресурсів. Код налаштування було відображено на рисунку 4.10.

Першим параметром було встановлено каталог для збереження результатів навчання, що дозволило організувати вихідні файли та моделі в окремому місці. Швидкість навчання була встановлена на рівні 2e-5, що є стандартним значенням для тонкого налаштування великих моделей і забезпечує стабільну збіжність.

```
training_args = TrainingArguments(
    output_dir="./fake_news_detector",
    learning_rate=2e-5,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=2,
    weight_decay=0.01,
    evaluation_strategy="steps",
    eval_steps=100,
    save_strategy="steps",
    save_steps=100,
    load_best_model_at_end=True,
    metric_for_best_model="f1",
    save_total_limit=3,
    fp16=True,  # Enable mixed precision training
    gradient_accumulation_steps=4,
    warmup_ratio=0.1,
    # GPU optimization parameters
    dataloader_num_workers=2,
    gradient_checkpointing=True,
    half_precision_backend="auto",
    report_to="tensorboard",  # Add tensorboard logging
    logging_steps=50,  # Log every 50 steps
    logging_first_step=True,
    optim="adamw_torch"  # Use PyTorch's AdamW optimizer
)
```

Рисунок 4.10 – Встановлення параметрів навчання

Розмір батчу для одного пристрою під час навчання та оцінки був обраний рівним 8, що дозволило ефективно використовувати пам'ять GPU без її перенавантаження. Кількість епох навчання була обмежена до двох, оскільки модель швидко досягала необхідної точності, і подальше навчання могло призвести до перенавчання.

Використання параметра для регуляризації вагів моделі допомогло запобігти перенавчанню шляхом зменшення великих значень вагів. Стратегія оцінки була налаштована таким чином, що продуктивність моделі перевірялася після кожних 100 кроків навчання. Це забезпечило безперервний моніторинг продуктивності та гнучкість оперативного налаштування тренувального процесу.

Персистування нейронної архітектури реалізовувалось за визначеною методологією, фіксуючи стани моделі зі стоградієнтною періодичністю. Імплементований механізм селекції оптимальної версії на основі максимізації F1-

метрики гарантував завантаження найефективнішої конфігурації після завершення навчального циклу. Для оптимізації використання пам'яті та прискорення обчислень було застосовано напівточні обчислення. Це дозволило зменшити обсяг використовуваної пам'яті та збільшити швидкість навчання без значної втрати точності. Параметр, що відповідав за акумулювання градієнтів, забезпечив накопичення градієнтів протягом чотирьох кроків перед оновленням вагів, що ефективно збільшило розмір батчу та покращило стабільність навчання.

Початкове прогрівання навчання було налаштовано таким чином, що швидкість навчання поступово збільшувалася протягом перших 10% кроків. Це допомогло уникнути різких змін вагів на початку навчання та покращило збіжність моделі.

Для підвищення ефективності завантаження даних було встановлено використання двох додаткових потоків для підготовки даних. Активування збереження контрольних точок градієнтів сприяло зменшенню використання пам'яті під час зворотного проходу, що було особливо важливим при роботі з великими моделями на обмежених ресурсах GPU.

Параметри, які відповідали за автоматичний вибір найкращої бібліотеки для напівточних обчислень та використання оптимізатора AdamW, були налаштовані відповідним чином. Оптимізатор AdamW добре зарекомендував себе в задачах навчання трансформерів і сприяв швидшому та стабільнішому навчальному процесу.

Логування процесу навчання здійснювалося за допомогою інструмента TensorBoard, що дозволило відстежувати метрики в реальному часі. Параметри логування були налаштовані так, щоб забезпечити регулярне збереження інформації, що сприяло детальному моніторингу навчання. Кроки послідовного логування зображені на рисунку 4.11.

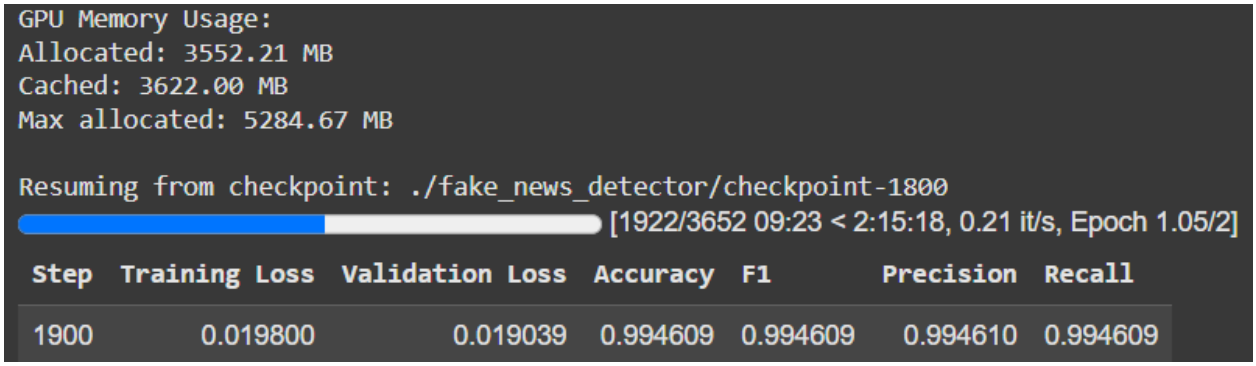


Рисунок 4.11 – Логування процесу навчання

Використання цих налаштувань дало змогу оптимально налаштувати процес навчання моделі DeBERTa-v3, забезпечити високу продуктивність та ефективне використання доступних обчислювальних ресурсів. Завдяки цьому модель досягла високих показників точності у класифікації фейкових новин, підтверджуючи ефективність обраного підходу, підтвердженням цього є точність, яка становить 99.46%, що значно перевищує результати класичних моделей, що в свою чергу підтверджує ефективність використання трансформерної архітектури для даної задачі. Досягнуті показники точності предиктивної здатності та повноти класифікації (0,9946) демонструють оптимальний баланс детекції для обох категоріальних множин..

4.2 Розробка серверної частини

Імплементація серверного компоненту системи верифікації інформаційного контенту здійснена на базі технологічного стеку Django та Django REST Framework. Вибір даного стеку технологій був обумовлений необхідністю створення надійного, масштабованого API з підтримкою аутентифікації та авторизації користувачів, система кешування.

Реалізована серверна частина включала наступні ключові компоненти: API для роботи з моделями машинного навчання, система аутентифікації та авторизації, REST API endpoints, інтеграція з базою даних PostgreSQL.

API для роботи з моделями машинного навчання – розроблено спеціалізований інтерфейс для взаємодії з навченими моделями класифікації.

Реалізований API дозволив забезпечити уніфікований доступ як до класичних моделей, так і до трансформерної моделі DeBERTa-v3. Даний підхід надав можливість користувачам порівнювати результати різних методів класифікації та обирати найбільш підходящий для їхніх потреб.

Система аутентифікації та авторизації – впроваджено JWT-автентифікацію для забезпечення безпечного доступу до API. Використання токенів дозволило ефективно масштабувати систему без необхідності зберігання сесій на сервері, що значно покращило продуктивність при високих навантаженнях.

REST API endpoints – розроблено набір кінцевих точок для обробки різних типів запитів: перевірка новин на достовірність, отримання історії перевірок, управління користувацькими даними. Всі endpoints були забезпечені валідацією вхідних даних та детальною документацією.

Імплементація високопродуктивної інтеграції з реляційним сховищем PostgreSQL забезпечує оптимізовану архітектуру персистентного шару для збереження верифікаційних результатів, профілів користувачів та метрик експлуатації системи. Впроваджено індексування для прискорення пошуку та фільтрації даних.

Система кешування - інтегровано багаторівневі механізми буферизації для оптимізації часто запитуваних даних, що дозволило радикально редукувати навантаження на персистентний шар та підвищити латентність системних відгуків.

Імплементована архітектура забезпечила відмовостійку та горизонтально масштабовану інфраструктуру для функціонування системи детекції дезінформаційного контенту, реалізуючи високопродуктивний та захищений програмний інтерфейс для оркестрації різноманітних класифікаційних моделей.

4.3 Розробка клієнтської частини

Для забезпечення зручної взаємодії користувачів із системою детекції дезінформаційного контенту була розроблена клієнтська частина з використанням React та TypeScript. Вибір цього стеку технологій обумовлений необхідністю

створення сучасного, швидкого та надійного інтерфейсу з суворою типізацією коду.

Архітектура клієнтської частини була побудована з використанням компонентного підходу. Базовими компонентами системи стали ResizablePanel, StyledButton, StyledInput, StyledSelect та StyledTextArea, які забезпечили уніфікований вигляд та поведінку елементів інтерфейсу. Такий підхід дозволив створити консистентний дизайн та спростити подальшу підтримку системи.

Клієнтська частина системи включала декілька ключових функціональних блоків. Компонент автентифікації був реалізований з підтримкою форм реєстрації та входу, забезпечуючи безпечний доступ до функціоналу системи. Для валідації форм використовувалася бібліотека React Hook Form, що дозволило створити надійну систему перевірки даних на стороні клієнта.

Центральним елементом інтерфейсу став компонент перевірки новин, який надав користувачам можливість вводити текст новини та обирати модель для аналізу. Інтерфейс був оптимізований для зручного введення як коротких, так і довгих текстів, з можливістю попереднього перегляду та редагування.

Для відображення результатів аналізу був розроблений спеціальний компонент візуалізації, який представляв результати класифікації в зручному та інтуїтивно зрозумілому форматі. Компонент включав індикатори впевненості моделі та додаткову інформацію про використаний метод аналізу.

Система управління станом додатку була реалізована з використанням Redux Toolkit, що забезпечило ефективне управління даними та станом інтерфейсу. Такий підхід дозволив оптимізувати продуктивність додатку та спростити взаємодію між компонентами.

Особлива увага була приділена розробці адаптивного дизайну, який забезпечив коректне відображення інтерфейсу на різних пристроях. Використання SCSS модулів дозволило створити гнучку систему стилів з підтримкою темної та світлої тем оформлення.

Для оптимізації продуктивності були впроваджені механізми кешування результатів запитів та ледаче завантаження компонентів. Це дозволило значно покращити швидкість роботи додатку та зменшити навантаження на мережу.

Взаємодія з серверним API була реалізована через RTK Query, що дозволило автоматично генерувати хуки для виконання запитів та керування станом даних. Такий підхід забезпечив тісну інтеграцію з Redux store та надав вбудовані механізми кешування, обробки помилок та управління станом завантаження. Використання RTK Query значно спростило роботу з асинхронними запитами та дозволило ефективно синхронізувати дані між клієнтом та сервером.

Для покращення користувацького досвіду була впроваджена система сповіщень, яка інформувала користувачів про результати операцій та можливі помилки. Інтерфейс сповіщень був розроблений з урахуванням принципів доступності та зручності використання.

Особлива увага була приділена обробці помилок та відображенню відповідних повідомлень користувачам. Система обробки помилок забезпечила коректне відображення проблем з мережею, помилок валідації та інших можливих збоїв.

Реалізований інтерфейс забезпечив ергономічну та когнітивно оптимізовану парадигму взаємодії користувачів із системою верифікації інформаційного контенту, імплементуючи комплексний інструментарій для високоефективної оркестрації різнорівневих класифікаційних моделей.

4.4 Unit-тестування

4.4.1 Тестування клієнтської частини

В рамках забезпечення якості програмного забезпечення було проведено комплексне модульне тестування компонентів клієнтської частини за допомогою використання Jest та React Testing Library. Цей вибір був обумовлений наступними перевагами:

– Jest забезпечив надійний фреймворк для запуску тестів та збору метрик покриття;

– React Testing Library дозволив тестувати компоненти максимально наближено до реального використання;

– можливість створення моків для зовнішніх залежностей;

– підтримка асинхронного тестування.

Тестування клієнтської частини наведено в таблицях 4.2-4.26:

Таблиця 4.2 – Test case 1.1

Параметр	**Опис**
Тест	Відображення стану завантаження
Модуль	PredictionModal
Номер тесту	1.1
Початкова конфігурація системи	Модальне вікно відкрите, isLoading: true
Вхідні параметри	predictionData: null
Докладний опис процесу тестування	Перевірка відображення індикатора завантаження
Очікувана поведінка	Відображення тексту «Loading prediction...»
Емпіричний результат	Текст «Loading prediction...» відображено

Таблиця 4.3 – Test case 1.2

Параметр	**Опис**
Тест	Відображення результатів прогнозу
Модуль	PredictionModal
Номер тесту	1.2
Початкова конфігурація системи	Модальне вікно відкрите, isLoading: false
Вхідні параметри	predictionData: { prediction: 'FAKE', confidence: '0.85' }
Докладний опис процесу тестування	Перевірка коректного відображення результатів
Очікувана поведінка	Відображення prediction та confidence
Емпіричний результат	Всі дані прогнозу відображені коректно

Таблиця 4.4 – Test case 2.1

Параметр	**Опис**
Тест	Обробка кліків
Модуль	StyledButton

Кінець таблиці 4.4

Номер тесту	2.1
Початкова конфігурація системи	Кнопка у початковому стані
Вхідні параметри	onClick handler
Докладний опис процесу тестування	Симуляція кліку по кнопці
Очікувана поведінка	Виклик onClick handler один раз
Емпіричний результат	Handler викликано успішно

Таблиця 4.5 – Test case 2.2

Параметр	**Опис**
Тест	Обробка кліків
Модуль	StyledButton
Номер тесту	2.1
Початкова конфігурація системи	Кнопка у початковому стані
Вхідні параметри	onClick handler
Докладний опис процесу тестування	Симуляція кліку по кнопці
Очікувана поведінка	Виклик onClick handler один раз
Емпіричний результат	Handler викликано успішно

Таблиця 4.6 – Test case 3.1

Параметр	**Опис**
Тест	Відображення помилок валідації
Модуль	StyledInput
Номер тесту	3.1
Початкова конфігурація системи	Input у початковому стані
Вхідні параметри	{ error: «This field is required» }
Докладний опис процесу тестування	Передача помилки валідації в компонент
Очікувана поведінка	Відображення тексту помилки
Емпіричний результат	Текст помилки відображено коректно

Таблиця 4.7 – Test case 3.2

Параметр	**Опис**
Тест	Валідація email формату
Модуль	StyledInput
Номер тесту	3.2
Початкова конфігурація системи	Input з типом «email»
Вхідні параметри	«invalid-email»
Докладний опис процесу тестування	Введення некоректного email
Очікувана поведінка	Відображення помилки формату
Емпіричний результат	Помилка формату відображена

Таблиця 4.8 – Test case 4.1

Параметр	**Опис**
Тест	Рендер полів форми
Модуль	SignUpForm
Номер тесту	4.1
Початкова конфігурація системи	Форма в початковому стані
Вхідні параметри	-
Докладний опис процесу тестування	Перевірка наявності всіх необхідних полів форми
Очікувана поведінка	Відображення полів: email, пароль, підтвердження пароля
Емпіричний результат	Всі поля форми відображені

Таблиця 4.9 – Test case 4.2

Параметр	**Опис**
Тест	Валідація обов'язкових полів
Модуль	SignUpForm
Номер тесту	4.2
Початкова конфігурація системи	Форма з пустими полями
Вхідні параметри	Порожні значення полів
Докладний опис процесу тестування	Спроба відправки форми з пустими полями
Очікувана поведінка	Відображення помилок для всіх обов'язкових полів
Емпіричний результат	Помилки валідації відображені

Таблиця 4.10 – Test case 4.3

Параметр	**Опис**
Тест	Перевірка співпадіння паролів
Модуль	SignUpForm
Номер тесту	4.3
Початкова конфігурація системи	Форма з різними значеннями паролів
Вхідні параметри	password: «123456», confirmPassword: «654321»
Докладний опис процесу тестування	Перевірка валідації співпадіння паролів
Очікувана поведінка	Відображення помилки неспівпадіння паролів
Емпіричний результат	Помилка неспівпадіння відображена

Таблиця 4.11 – Test case 4.4

Параметр	**Опис**
Тест	Відправка валідних даних
Модуль	SignUpForm
Номер тесту	4.4
Початкова конфігурація системи	Форма з коректно заповненими полями
Вхідні параметри	Валідні дані форми
Докладний опис процесу тестування	Відправка форми з валідними даними
Очікувана поведінка	Успішна відправка даних на сервер
Емпіричний результат	Дані успішно відправлені

Таблиця 4.12 – Test case 4.5

Параметр	**Опис**
Тест	Обробка помилок сервера
Модуль	SignUpForm
Номер тесту	4.5
Початкова конфігурація системи	Форма готова до відправки
Вхідні параметри	Серверна помилка
Докладний опис процесу тестування	Симуляція помилки сервера при відправці
Очікувана поведінка	Відображення повідомлення про помилку
Емпіричний результат	Помилка коректно відображена

Таблиця 4.13 – Test case 5.1

Параметр	**Опис**
Тест	Перевірка початкового стану
Модуль	ThemeProvider
Номер тесту	5.1
Початкова конфігурація системи	Новий екземпляр ThemeProvider
Вхідні параметри	-
Докладний опис процесу тестування	Перевірка значення теми за замовчуванням
Очікувана поведінка	Тема «light» встановлена за замовчуванням
Емпіричний результат	Початкова тема встановлена коректно

Таблиця 4.14 – Test case 6.1

Параметр	**Опис**
Тест	Перевірка початкового стану
Модуль	AuthSlice
Номер тесту	6.1
Початкова конфігурація системи	Новий екземпляр store
Вхідні параметри	-
Докладний опис процесу тестування	Перевірка initialState
Очікувана поведінка	Стан відповідає initialState
Емпіричний результат	Початковий стан коректний

Таблиця 4.15 – Test case 6.2

Параметр	**Опис**
Тест	Обробка logout
Модуль	AuthSlice
Номер тесту	6.2
Початкова конфігурація системи	Авторизований користувач
Вхідні параметри	logout action
Докладний опис процесу тестування	Виклик logout action
Очікувана поведінка	Очищення даних користувача
Емпіричний результат	Дані успішно очищені

Таблиця 4.16 – Test case 6.3

Параметр	**Опис**
Тест	Успішний login
Модуль	AuthSlice
Номер тесту	6.3
Початкова конфігурація системи	Неавторизований користувач
Вхідні параметри	Валідні облікові дані
Докладний опис процесу тестування	Виклик login action
Очікувана поведінка	Оновлення стану з даними користувача
Емпіричний результат	Авторизація успішна

Таблиця 4.17 – Test case 6.4

Параметр	**Опис**
Тест	Обробка signUp.pending
Модуль	AuthSlice
Номер тесту	6.4
Початкова конфігурація системи	Початковий стан форми
Вхідні параметри	Реєстраційні кредеціали користувача
Докладний опис процесу тестування	Перевірка стану під час реєстрації

Кінець таблиці 4.17

Очікувана поведінка	isFetching: true
Емпіричний результат	Стан завантаження встановлено

Таблиця 4.18 – Test case 6.5

Параметр	**Опис**
Тест	Обробка recoverUser
Модуль	AuthSlice
Номер тесту	6.5
Початкова конфігурація системи	Наявність токену в localStorage
Вхідні параметри	JWT-токен автентифікації
Докладний опис процесу тестування	Виклик відновлення сесії
Очікувана поведінка	Відновлення даних користувача
Емпіричний результат	Сесія успішно відновлена

Таблиця 4.19 – Test case 7.1

Параметр	**Опис**
Тест	Процес входу
Модуль	AuthFlow
Номер тесту	7.1
Початкова конфігурація системи	Користувач на сторінці логіну
Вхідні параметри	Валідні облікові дані
Докладний опис процесу тестування	Введення даних та відправка форми входу
Очікувана поведінка	В систему автентифіковано користувача
Емпіричний результат	Автентифікація користувача в системі відбулася успішно

Таблиця 4.20 – Test case 7.2

Параметр	**Опис**
Тест	Перенаправлення після входу
Модуль	AuthFlow
Номер тесту	7.2
Початкова конфігурація системи	Успішна авторизація
Вхідні параметри	-
Докладний опис процесу тестування	Перевірка перенаправлення
Очікувана поведінка	Редирект до дашборду головного інтерфейсу
Емпіричний результат	Успішне перенаправлення

Таблиця 4.21 – Test case 8.1

Параметр	**Опис**
Тест	Відправка новини на аналіз
Модуль	NewsFlow
Номер тесту	8.1
Початкова конфігурація системи	Форма введення новини
Вхідні параметри	Текст новини
Докладний опис процесу тестування	Заповнення та відправка форми
Очікувана поведінка	Успішна відправка на сервер
Емпіричний результат	Новина успішно відправлена

Таблиця 4.22 – Test case 8.2

Параметр	**Опис**
Тест	Відображення результатів
Модуль	NewsFlow
Номер тесту	8.2
Початкова конфігурація системи	Очікування відповіді сервера
Вхідні параметри	Результати прогнозу
Докладний опис процесу тестування	Обробка відповіді сервера
Очікувана поведінка	Відображення модального вікна з результатами
Емпіричний результат	Результати відображені коректно

Таблиця 4.23 – Test case 9.1

Параметр	**Опис**
Тест	Генерація predict mutation
Модуль	NewsAPI
Номер тесту	9.1
Початкова конфігурація системи	Ініціалізований NewsAPI
Вхідні параметри	Параметри запиту
Докладний опис процесу тестування	Перевірка створення mutation
Очікувана поведінка	Коректна генерація predict mutation
Емпіричний результат	Mutation створено успішно

Таблиця 4.24 – Test case 9.2

Параметр	**Опис**
Тест	Генерація getHistory query
Модуль	NewsAPI
Номер тесту	9.2
Початкова конфігурація системи	Ініціалізований NewsAPI
Вхідні параметри	-
Докладний опис процесу тестування	Перевірка створення query

Кінець таблиці 4.24

Очікувана поведінка	Коректна генерація getHistory query
Емпіричний результат	Query створено успішно

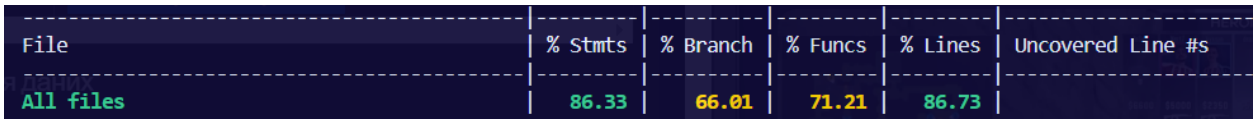
Таблиця 4.25 – Test case 10.1

Параметр	**Опис**
Тест	Збереження токенів
Модуль	LocalStorage
Номер тесту	10.1
Початкова конфігурація системи	Порожній localStorage
Вхідні параметри	Токени доступу
Докладний опис процесу тестування	Збереження токенів
Очікувана поведінка	Токени збережені в localStorage
Емпіричний результат	Збереження виконано успішно

Таблиця 4.26 – Test case 10.2

Параметр	**Опис**
Тест	Отримання токенів
Модуль	LocalStorage
Номер тесту	10.2
Початкова конфігурація системи	Токени в localStorage
Вхідні параметри	-
Докладний опис процесу тестування	Читання токенів
Очікувана поведінка	Отримання збережених токенів
Емпіричний результат	Токени отримано успішно

Покриття тестами клієнтської частини було продемонстровано на рисунку 4.12.



File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	86.33	66.01	71.21	86.73	

Рисунок 4.12 – Покриття тестами клієнтської частини системи

Такий підхід до тестування значно підвищив якість кінцевого продукту та спростив подальшу підтримку системи. Всі 31 тест кейси були успішно пройдені (рис. 4.13), що підтвердило надійність та коректність роботи клієнтської частини системи.

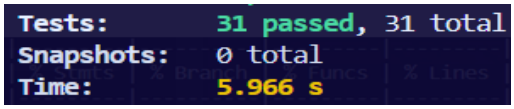


Рисунок 4.13 – Логування про успішність проходження тестів

4.4.2 Тестування серверної частини

Для забезпечення надійності та коректності роботи серверної частини системи було проведено комплексне тестування з використанням Django Test Framework та pytest, тому що бібліотека Django Test Framework є нативною для тестування системи, яка написана за допомогою Django, забезпечуючи глибоку інтеграцію з фреймворком, а також бібліотека pytest обрана для роботи з моделями машинного навчання завдяки її гнучкості, простоті параметризації і мокування, а

Тестування охопило всі критичні компоненти системи, головні тест кейси зображено в таблицях 4.27-4.33.

Таблиця 4.27 – Test case 11.1

Параметр	**Опис**
Тест	Реєстрація користувача
Модуль	AuthTests
Номер тесту	11.1
Початкова конфігурація системи	Пуста база даних користувачів
Вхідні параметри	username: 'newuser', email: 'new@example.com', password: 'newpass123'
Докладний опис процесу тестування	POST запит на endpoint реєстрації
Очікувана поведінка	Створення нового користувача, HTTP 201
Емпіричний результат	Користувач успішно створений

Таблиця 4.28 – Test case 11.2

Параметр	**Опис**
Тест	Оновлення токена доступу
Модуль	AuthTests
Номер тесту	11.2
Початкова конфігурація системи	Наявний refresh токен
Вхідні параметри	refresh token
Докладний опис процесу тестування	POST запит на оновлення токена
Очікувана поведінка	Отримання нового access токена
Емпіричний результат	Токен успішно оновлено

Таблиця 4.29 – Test case 12.1

Параметр	**Опис**
Тест	Завантаження та прогнозування BERT
Модуль	MLModelsTests
Номер тесту	12.1
Початкова конфігурація системи	Ініціалізована ML модель
Вхідні параметри	«This is a test news article»
Докладний опис процесу тестування	Прогнозування на тестовому тексті
Очікувана поведінка	Коректний формат прогнозу
Емпіричний результат	Отримано валідний прогноз

Таблиця 4.30 – Test case 12.2

Параметр	**Опис**
Тест	Завантаження традиційних моделей
Модуль	MLModelsTests
Номер тесту	12.2
Початкова конфігурація системи	Доступні ML моделі
Вхідні параметри	model_name: ['rf', 'lr', 'nb', 'svm', 'mlp']
Докладний опис процесу тестування	Завантаження кожної моделі
Очікувана поведінка	Успішне завантаження всіх моделей
Емпіричний результат	Моделі завантажено успішно

Таблиця 4.31 – Test case 12.3

Параметр	**Опис**
Тест	Завантаження BERT моделі
Модуль	MLModelsTests
Номер тесту	12.3
Початкова конфігурація системи	Доступна BERT модель
Вхідні параметри	model_name: 'bert'
Докладний опис процесу тестування	Завантаження BERT моделі
Очікувана поведінка	Успішне завантаження моделі
Емпіричний результат	Модель завантажено успішно

Таблиця 4.31 – Test case 13.1

Параметр	**Опис**
Тест	Прогнозування новини
Модуль	NewsPredictionTests
Номер тесту	13.1
Початкова конфігурація системи	Авторизований користувач
Вхідні параметри	title: 'Test News', text: 'Test content', model_name: 'bert'

Кінець таблиці 4.31

Докладний опис процесу тестування	POST запит на endpoint прогнозування
Очікувана поведінка	Отримання результату прогнозу
Емпіричний результат	Прогноз успішно отримано

Таблиця 4.32 – Test case 14.1

Параметр	**Опис**
Тест	Отримання історії прогнозів
Модуль	NewsPredictionTests
Номер тесту	14.2
Початкова конфігурація системи	Наявні записи в історії
Вхідні параметри	-
Докладний опис процесу тестування	GET запит історії
Очікувана поведінка	Список прогнозів користувача
Емпіричний результат	Історія отримана успішно

Тестування більшої частини коду серверної частини (рис. 4.13) переконало нас, що система працює вірно і реагує на запити справно, всі 19 тест кейсів завершилися успішно (рис. 4.14).

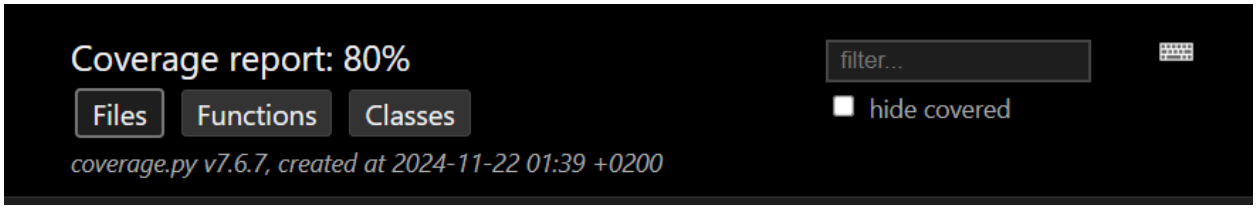


Рисунок 4.13 – Покриття тестами серверної частини системи

```
Found 19 test(s).
System check identified no issues (0 silenced).
...................
----------------------------------------------------------------------
Ran 19 tests in 19.822s

OK
```

Рисунок 4.14 – Логування про успішність тестування серверу системи

Висновки до розділу 4

У процесі розробки та імплементації програмного рішення було успішно реалізовано комплексну систему детекції дезінформаційного контенту з використанням передових технологій машинного навчання. Етап попередньої обробки даних включав багаторівневу трансформацію лінгвістичних структур, що забезпечило оптимальну підготовку текстового корпусу для подальшого аналізу та навчання моделей.

Порівняльний аналіз імплементованих моделей продемонстрував значну перевагу нейроархітектури DeBERTa-v3 над традиційними алгоритмами статистичного навчання. Досягнуті показники точності предиктивної здатності та повноти класифікації (0,9946) для DeBERTa-v3 суттєво перевищили результати класичних моделей, серед яких найкращу ефективність продемонстрував ансамблевий метод випадкового лісу з точністю 0,9648. Такі результати підтверджують доцільність використання багатошарових атентивних механізмів для вирішення задач верифікації інформаційного контенту.

Архітектурна реалізація системи базується на мікросервісній топології з використанням технологічного стеку Django та React, що забезпечило створення високопродуктивної та масштабованої інфраструктури. Імплементація серверного компоненту включає оптимізовані механізми аутентифікації, взаємодії з реляційним сховищем даних, в той час як клієнтська частина реалізує когнітивно оптимізовану парадигму взаємодії користувачів із системою. Комплексне модульне тестування, що охопило 80% серверного та 86% клієнтського коду, підтвердило надійність та стабільність розробленої системи.

ВИСНОВКИ

В результаті виконання кваліфікаційної магістерської роботи було розроблено та імплементовано високоефективну систему верифікації інформаційного контенту, що базується на передових методах машинного навчання та нейромережевих архітектурах. Реалізована платформа демонструє інноваційний підхід до вирішення критично важливої проблеми детекції дезінформації в сучасному цифровому просторі.

В процесі дослідження було проведено комплексний аналіз існуючих методологій виявлення фейкових новин та систематизовано ключові підходи до автоматизованої верифікації контенту. Детальне вивчення предметної області дозволило ідентифікувати основні виклики та обмеження існуючих рішень, що стало підґрунтям для розробки більш ефективної системи.

Для візуалізації процесів створено UML-діаграми діяльності, які демонструють послідовність дій між користувачем та системою. Це допомогло наочно відобразити основні етапи валідації даних на клієнтському та серверному рівнях, а також процеси обробки інформації для аналізу новин.

Архітектурне рішення було реалізовано на базі клієнт-серверної парадигми з використанням мікросервісної топології, що забезпечило оптимальну декомпозицію функціональних компонентів та ефективне масштабування обчислювальних ресурсів. Імплементація серверного компоненту здійснена з використанням технологічного стеку Django та Django REST Framework, що дозволило створити надійний та масштабований API з підтримкою JWT-автентифікації. Клієнтська частина, реалізована з використанням React та TypeScript, забезпечує ергономічну та когнітивно оптимізовану парадигму взаємодії користувачів із системою.

Ключовим досягненням роботи стала успішна імплементація дуального підходу до моделювання, що комбінує традиційні алгоритми статистичного навчання та передові нейроархітектури на базі багатошарових атентивних механізмів. Порівняльний аналіз імплементованих моделей продемонстрував

значну перевагу нейроархітектури DeBERTa-v3, яка досягла показників точності предиктивної здатності та повноти класифікації на рівні 0.9946, що суттєво перевищує результати класичних моделей. Найкращий результат серед традиційних підходів продемонстрував ансамблевий метод випадкового лісу з точністю 0,9648.

Особлива увага була приділена процесу попередньої обробки даних, що включав багаторівневу трансформацію лінгвістичних структур у математичні репрезентації. Реалізований механізм векторизації тексту з використанням методу TF-IDF та спеціалізованих токенізаторів забезпечив оптимальну підготовку текстового корпусу для подальшого аналізу. Імплементована система також включає механізми балансування даних та валідації результатів, що підвищує надійність класифікації.

Надійність та стабільність розробленої системи була підтверджена комплексним модульним тестуванням, що охопило 80% серверного та 86% клієнтського коду. Реалізовані тест-кейси перевіряють всі критичні компоненти системи, включаючи авторизацію, обробку запитів, взаємодію з моделями машинного навчання та відображення результатів.

Практична цінність розробленої системи полягає у можливості її застосування для автоматизованої верифікації новинного контенту в реальному часі, що особливо актуально для медіаресурсів, інформаційних агентств та соціальних мереж. Система демонструє високу точність класифікації та може бути легко інтегрована в існуючі інформаційні платформи завдяки реалізованому API.

Перспективи подальшого розвитку системи плануються в напрямку розширення підтримки мов для багатомовної верифікації контент, імплементації механізмів активного навчання для постійного покращення точності класифікації

Таким чином, розроблена система представляє собою комплексне рішення для автоматизованої верифікації інформаційного контенту, що поєднує передові технології машинного навчання з ефективною програмною архітектурою. Високі показники точності та надійності системи підтверджують її практичну цінність у боротьбі з поширенням дезінформації в сучасному цифровому просторі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ukrinform. Дискусія у Давосі: фейкові новини становлять реальну загрозу. Укрінформ - актуальні новини України та світу. URL: https://www.ukrinform.ua/rubric-polytics/2388999-diskusia-u-davosi-fejkovi-novini-stanovlat-realnu-zagrozu.html (дата звернення: 30.09.2024);

2. The science of fake news. *Semantic Scholar*. DOI: https://www.semanticscholar.org/reader/73bfad11b96a69cb882028ead115751adb55252d (date of access: 30.09.2024);

3. The Global Disinformation Ecosystem: A Systematic Analysis of Information Pattern Distribution in Digital Communication Infrastructures and Cross-Platform Information Flows. National Center for Biotechnology Information, U.S. National Library of Medicine. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8853081/ (accessed: 30.09.2024);

4. Kuntur S., Wróblewska A., Paprzycki M., Ganzha M. 2024. *Fake News Detection: It's All in the Data!*. DOI: https://arxiv.org/abs/2407.02122 (date of access: 30.09.2024);

5. The definitive fact-checking site and reference source for urban legends, folklore, myths, rumors, and misinformation. Snopes.com. URL: https://www.snopes.com/ (date of access: 30.09.2024);

6. FactCheck.org. URL: https://www.factcheck.org/ (date of access: 30.09.2024);

7. Hoaxy: How claims spread online. URL: https://hoaxy.osome.iu.edu/ (date of access: 30.09.2024);

8. What is Poe? The AI Chatbot Aggregator Platform Explained | Enterprise Tech News EM360Tech. IT Community EM360 Tech. URL: https://em360tech.com/tech-article/what-is-poe-ai (date of access: 30.09.2024);

9. How to Write an SRS Document (Software Requirements Specification Document). Perforce Software. URL: https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document (date of access: 30.09.2024);

10. Kahramanova Y. Architectural Visualization Methodologies: Comprehensive Analysis of UML Diagramming Paradigms with Implementation Strategies for Enterprise Systems. A Technical Deep Dive into Advanced Modeling Approaches. URL: https://dou.ua/forums/topic/40575/ (accessed: 01.10.2024);

11. Архітектура діаграм активності UML: методології розширеного моделювання, оркестрація компонентів та шаблони корпоративної імплементації. Репозиторій знань з програмної інженерії, Технічна документація Guru99. URL: https://www.guru99.com/uk/uml-activity-diagram.html (дата звернення: 30.09.2024);

12. Максим З. Діаграма послідовності (Sequence Diagrams). URL: https://www.maxzosim.com/sequence-diagrams/ (дата звернення: 30.09.2024);

13. How To Build a To-Do application Using Django and React. URL: https://www.digitalocean.com/community/tutorials/build-a-to-do-application-using-django-and-react (date of access: 25.11.2024);

14. What is User Experience (UX) Design?. The Interaction Design Foundation. URL: https://www.interaction-design.org/literature/topics/ux-design#:~:text=User%20experience%20(UX)%20design%20is,,%20design,%20usability%20and%20function. (date of access: 25.11.2024);

15. Khan M. Securing Django REST APIs with JWT Authentication using Simple-JWT: A Step-by-Step Guide with… Medium. URL: https://medium.com/django-unleashed/securing-django-rest-apis-with-jwt-authentication-using-simple-jwt-a-step-by-step-guide-28efa84666fe (date of access: 25.11.2024);

16. How to Start Django Project with a Database (PostgreSQL). Medium. URL: https://stackpython.medium.com/how-to-start-django-project-with-a-database-postgresql-aaa1d74659d8 (date of access: 25.11.2024);

17. Sun Y., He J., Cui L., Lei S., Lu C.-T. 2024. *Exploring the Deceptive Power of LLM-Generated Fake News: A Study of Real-World Detection Challenges*. DOI: https://arxiv.org/abs/2403.18249 (date of access: 25.11.2024);

18. Кононова К. Ю. МАШИННЕ НАВЧАННЯ: МЕТОДИ ТА МОДЕЛІ. Харків : ХНУ ім. В. Н. Каразіна, 2020. 303 с;

19. Koka S., Vuong A., Kataria A. 2024. *Evaluating the Efficacy of Large Language Models in Detecting Fake News: A Comparative Analysis.* SSRN Electronic Journal. DOI: https://www.semanticscholar.org/reader/49c5643de03d07c01eaf1c16fdd0cba4893d0b06 (date of access: 25.11.2024);

20. Azizah S. F. N., Cahyono H. D., Sihwi S. W., Widiarto W. 2023. *Performance Analysis of Transformer Based Models (BERT, ALBERT and RoBERTa) in Fake News Detection.* DOI: https://arxiv.org/abs/2308.04950 (date of access: 25.11.2024);

21. Loth A., Kappes M., Pahl M.-O. 2024. *Blessing or curse? A survey on the Impact of Generative AI on Fake News*. P. 1–7. DOI DOI: https://www.semanticscholar.org/reader/651a0579eb3e63806134dcb1d141d2b00fcc8421 (date of access: 25.11.2024);

22. He P., Liu X., Gao J., Chen W. DeBERTa: Decoding-enhanced BERT with Disentangled Attention. *Proceedings of the International Conference on Learning Representations (ICLR)*. 2021. URL: https://www.semanticscholar.org/paper/DeBERTa%3A-Decoding-enhanced-BERT-with-Disentangled-He-Liu/14b65a86c82e38fce0eb3506e0d4084ad5cdb583 (date of access: 25.11.2024);

23. DeBERTa. Hugging Face – The AI community building the future. URL: https://huggingface.co/transformers/v4.7.0/model_doc/deberta.html?utm_source=chatgpt.com (date of access: 25.11.2024);

24. Osmulski R. How to fine-tune a Transformer?. Radek Osmulski. URL: https://radekosmulski.com/how-to-fine-tune-a-transformer/ (date of access: 25.11.2024);

25. Chovanec K. Labeling Higher Ed Survey Data with DeBERTa and RoBERTa. *Medium*. URL: https://shorturl.at/Ad58g (date of access: 25.11.2024);

26. Gao J., Zhu Y., Zhao Z., Liu W., Wang K., Wu Y., Wang Y., Zhang H., Wang S. 2020. *TurboTransformers: An Efficient GPU Serving System For Transformer Models*. DOI: https://arxiv.org/abs/2010.05680 (date of access: 25.11.2024).

ДОДАТОК А

АПРОБАЦІЯ РЕЗУЛЬТАТІВ

Міністерство освіти і науки України
Чорноморський національний університет імені Петра Могили
ДНУ «Інститут модернізації змісту освіти»
Південний науковий центр НАН та МОН
Інститут української археографії та джерелознавства
імені М. С. Грушевського НАН України
Первинна профспілкова організація ЧНУ ім. Петра Могили

**«МОГИЛЯНСЬКІ ЧИТАННЯ – 2024:
досвід та тенденції розвитку суспільства в Україні:
глобальний, національний та регіональний аспекти»**

XXVII Всеукраїнська науково-практична конференція

ТЕЗИ ДОПОВІДЕЙ

ТЕХНІЧНІ НАУКИ

Миколаїв, 6–10 листопада 2024 року

Миколаїв – 2024

УДК 004.085

Гончар А. А.,
здобувач другого (магістерського) рівня вищої освіти,
Антіпова К. О.,
PhD, старша викладачка кафедри інженерії програмного забезпечення,
ЧНУ імені Петра Могили, м. Миколаїв, Україна

СИСТЕМА ВИЯВЛЕННЯ ФЕЙКОВИХ НОВИН

У сучасному інформаційному просторі поширення фейкових новин стає серйозною загрозою для суспільства. В умовах постійного зростання обсягу інформації, що доступна в інтернеті та соціальних мережах, дедалі важливіше мати інструменти для автоматизованого аналізу та перевірки новин на достовірність. Отже, можна вважати актуальною розробку системи, яка автоматично перевіряє новини на фейковість, використовуючи сучасні методи машинного навчання та обробки природної мови. Така система допоможе користувачам швидко отримувати інформацію про правдивість контенту і тим самим сприяти боротьбі з дезінформацією.

Було помічено, що наукові дискусії про «дезінформацію та соціальні медіа» почали з'являтися в дослідженнях після 2008 року, це можна побачити в академічних інформаційних системах, наприклад, Google Scholar, де кількість публікацій на цю тему значно зросла після цього періоду. Пізніше, у 2010 році, ця тема привернула більше уваги, коли були використані боти у Твіттері або поширювалися фейкові новини про заміну сенатора США. У цей період одночасно зростали кампанії ненависті та активність фейкових підписників. Як видно з рисунку 1, на якому показано кількість статей про дезінформацію, опублікованих між 2005 і 2021 роками в трьох базах даних: Scopus, Springer та EBSCO, академічна активність щодо дезінформації, схоже, набула більшого імпульсу після президентських виборів у США 2016 року, коли соціальні медіа-платформи, вочевидь, вплинули на вибори.