**ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ**

| IDE | – | Integrated Development Environment – інтегроване середовище розробки. |
| --- | --- | --- |
| API | – | Application programming interface, прикладний програмний Інтерфейс. |
| SDK | – | Software development kit. |
| IT | – | Інформаційні технології. |
| ER | – | Entity-Relation diagram. |
| OC | – | Операційна система. |
| БД  LLM  RSS  NLP  GPT | –  –  –  –  – | База даних.  Велика мовна модель  Веб-стрічка новин  Обробка природньої мови  Генеративний натренований трансформер |

**ВСТУП**

У сучасному суспільстві 21 століття питання надійності новинного контенту набуло особливої актуальності. Збільшення кількості фейків у щоденному інформаційному просторі становить серйозну загрозу для країн та її громадян, особливо в умовах збройного конфлікту[1]. В Україні, де інфопростір зазнає значного тиску внаслідок війни, потреба в засобах для перевірки фактів та аналізу контенту сильно зросла [2].

Можливі розв’язки таких задач включають використання:

* Алгоритми глибоких нейронних мереж, зокрема LLM для аналізу оцінки статей за метриками якості а також автоматизованої категоризації текстів для пошуку закономірностей;
* Семантичний пошук з використанням засобів векторизації тексту для пошуку схожих за значенням текстів [3];
* Використання розслідувань і статей від відомих і визнаних організацій факт-чекерів, наприклад VoxUkraine[4] & StopFake[5];

Задача цього проекту - створення веб-застосунку для агрегації та інтелектуального аналізу новинного контенту, який інтегруватиме сучасні технології штучного інтелекту для автоматизованої оцінки якості новин за критеріями достовірності , фактичності, емоційності та сенсаційності.

Метою проекту є покращення доступу користувачів до достовірної інформації шляхом створення системи автоматизованого контролю якості новинного контенту з використанням великих мовних моделей і аналізом природної мови.

# ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

## Постановка завдання дипломного проєктування

Як було зазначено у вступі, даний проєкт націлений на використання сучасних засобів штучного інтелекту та алгоритмів обробки природної мови для аналізу новинних статей від декількох джерел новинних статей, зокрема від таких новинних видань: espreso.tv[6], hromadske[7], ukrinform[8], Радіо «Свобода»[9]. Збір контенту можна виконувати за допомогою багаточисельних технологій веб-скрейпінгу або API новин.

Використання вище зазначених технологій відкриває наступні можливості у сфері аналізу новинного контенту

* Оцінка якості новинних статей за деякими метриками;
* Фільтрування статей за вищевказаними метриками або за видавниками статей;
* Категоризація розслідувань від факт-чекінгових організацій для визначення трендів дезінформації;
* Пошук схожих статей за тематикою і сенсом;
* Створення звітів про дезінформацію за деякий проміжок часу;

Реалізація даних завдань надасть можливість створити зрозумілий і простий інтерфейс для користувачів для доступу до цього функціоналу.

## Аналіз предметної області

За даними міжнародної PR-кампанії Edelman у її розслідуванні “Edelman Trust Barometer” 2023 року, в Об’єднаному Королівстві лише 37% громадян довіряють традиційним медіа [10], що свідчить про кризу довіри до журналістів.

Також варто зазначити, що український медіа-простір все більше заповнюється фейками. З найвидатніших – це підробка фактів про «біо-лабораторії США» [11], дезінформація про події в Бучі в 2022 році [12], і також не можна не згадати багаточисельні фейки про пошкодження інфраструктури через масовані російські обстріли [13].

Через такі кампанії зростає попит не тільки на якісну журналістику, а й на незалежних факт-чекерів, та можливість розрізняти якісну журналістику від неякісної.

Тим не менш, в інтернеті існує велика кількість онлайн-сервісів індивідуальних видань, де користувачі можуть подивитися їхні публікації, проте їхня проблема полягає у відстутності інтеграції з сучасними технологіями для аналізу тексту. Також велика кількість подібних сервісів мають суб'єктивні погляди і намагаються проштовхнути власні корисні для них наративи.

За останні роки також почали з'являтися агрегатори новин з метою подолання проблем, які виникають при обмеженні сприйняття новинного контенту від лише індивідуальних організацій.

## Аналіз існуючих рішень

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації моєї власної розробки веб-застосунку «NewsCheck». Далі будуть розглянуті готові програмні рішення, допоміжні програмні засоби та засоби розробки.

### Аналіз відомих програмних продуктів

Проведемо аналіз існуючих агрегаторів новин:

* Google News [14]: агрегатор новин від корпорації Google. Спеціалізується на агрегації новинного контенту від великої кількості організацій, та використовує кластеризацію для пошуку схожих статей.
* Ground News [15]: агрегатор новин, який використовує інновативний підхід для показу різних поглядів на одну подію від декількох видань.
* NewsGuard [16]: агрегатор новин, який використовує експертний підхід до оцінки достовірності сайтів медіа.

Для порівняння проєкту з аналогом можна скористатись таблицею 1.3.

Таблиця 1.3 – Порівняння з аналогами

| Функціонал | News  Check | Ground News | Google News | News Guard | Пояснення |
| --- | --- | --- | --- | --- | --- |
| Українські джерела новин | + | +- | + | +- | Google News підтримує українські джерела, Ground News лише висвітлює події України від міжнародних медіа, а NewsGuard має рейтинг для малої кількості українських медіа |
| Використання LLM для аналізу новин | + | - | - | - | Жодний з сервісів не використовує LLM для аналізу новин |
| Пошук схожих статей за семантичним пошуком | + | +- | +- | - | Google News і Ground News мають базовий пошук по ключовим словам і тематиці. |
| Інтеграція факт-чекерів | + | - | - | - | Жодний з сервісів не мають інтеграції з  факт-чекерами |
| Опис дезінормаційних кампаній | + | - | - | - | Жодний з сервісів не має опису дезінформаційних кампаній |

### Аналіз відомих алгоритмічних та технічних рішень

Для виконання функціональних вимог проєкту, варто проаналізувати технічні і алгоритмічні рішення, які є доступними.

1. Збір новинного контенту

Для збору новинного контенту, є наступні опції:

* 1. Статичний скрапінг: використання бібліотек для отримання вихідного коду веб-сторінки для подальшого збору
  2. Динамічний скрапінг: використання веб-кролерів, тобто емуляції браузера, для можливості взаємодії з JavaScript для кращої взаємодії зі сторінкою
  3. Гібридний підхід: використання обох попередніх підходів в залежності від сайту

Було прийнято рішення обрати підхід с.

1. Алгоритми векторизації тексту:
   1. Використання власної невеликої моделі, наприклад TF-IDF. Є швидкодійним варіантом, проте потрібно проводити тренування
   2. Використання більшої моделі, розгорнутої локально, наприклад BERT. Є кращим варіантом, але потребує великих обчислювальних ресурсів.
   3. Використання моделей для ембедингу на основі GPT. Має найкращі семантичні здібності, але потребує колосальних обчислювальних ресурсів, або використання API компаній VoyageAI або OpenAI, що збільшує витрати.

Було прийнято рішення обрати варіант с. з використанням API VoyageAI

1. Алгоритми категоризації статей факт-чекерів:
   1. Локальна модель машинного навчання або невелика нейронна мережа. Ефективний варіант, але потребує тренування, і також більше вразливий до зроблення помилок.
   2. Надсилання запитів для категоризації до LLM, наприклад OpenAI, що збільшує витрати.

Було прийнято рішення використати варіант b.

1. Оцінка статті за метриками якості:

Було прийнято рішення використати запити до LLM від OpenAI.

1. Зберігання векторів:
   1. Налаштування бази даних векторів, наприклад Milvus, з локальним розгортанням
   2. Використання хмарного серверного векторного сховища, наприклад Pinecone

Враховуючи простоту локального розгортання та зручність використання, було обрано Milvus

## Аналіз та моделювання бізнес-процесів

Для моделювання бізнес-процесу використовується BPMN модель (рисунок 1.1, рисунок 1.2, рисунок 1.3):

Опис послідовності аналізу статті новин:

1. Відбувається виклик завдання на аналіз статей
2. Дані збираються методом гібридного веб-скрейпінгу
3. Дані зберігаються у базі даних
4. Надсилається запит до LLM для оцінки статті
5. Метрики оцінки зберігаються у базі даних
6. Текст статті відправляється на векторизацію
7. Вектор статті зберігається у векторній базі даних
8. Виконується пошук схожих статей за допомогою семантичного пошуку
9. Результати пошуку зберігаються у базі даних

A diagram of a company

AI-generated content may be incorrect.

Рисунок 1.1 – Модель бізнес-процесу для збору і аналізу новинних статей

Опис послідовності аналізу розслідувань:

1. Викликається задача аналізу розслідувань
2. Проводиться збір методом веб-скрапінгу
3. Дані розслідування зберігаються у базі даних
4. Відправляється запит до LLM для категоризації статті
5. Результати аналізу зберігаються у базі даних

A diagram of a company

AI-generated content may be incorrect.

Рисунок 1.2 – Модель бізнес процесу для збору і аналізу розслідувань

Опис послідовності створення щотижневої статистики і дайджесту про дезінформацію:

1. Викликається задача для щотижневого аналізу
2. З бази даних агрегуються усі розслідування за минулий тиждень
3. Підраховуються усі кількість розслідувань для кожної категорії для статистики
4. До LLM надсилається запит для створення дайджесту
5. Створений дайджест і статистика зберігаються у базі даних

A diagram of a company

AI-generated content may be incorrect.

Рис. 1.3 – Модель бізнес-процесу для щотижнеового аналізу дезінформації

## Висновки до розділу

У цьому розділі було виконане детальне обстеження предметної області.

У першому підрозділі було наведено аргументацію щодо актуальності розробки цього проєкту, наведено технології, якими можна втілити функціонал проєкту.

У другому підрозділі було проведено ретельний аналіз предметної області, було наведено загальну теорію і контекст, що оточує дану предметну область. Було також представлено проблеми, з якими стикаються сучасні сервіси для представлення новинного контенту.

На початку третього підрозділу, було проведено аналіз існуючих рішень та існуючих програмних продуктів, складено таблицю для порівняння функціоналу власного проєкту і вище вказаних програмних продуктів. По результатам таблиці можна стверджувати, що функціонал і технології, які використовуються у проєкті є дійсно інноваційними. Далі було проведено аналіз існуючих технологій та алгоритмів, які можна було б використовувати для втілення програмного забезпечення проєкту.

У четвертому підрозділі було наведено опис бізнес-процесів, а також було створено нотації моделей бізнес-процесів за допомогою діаграм BPMN. Було узагальнено мету і цілі програмного забезпечення.

# РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Варіанти використання програмного забезпечення

Головним функціоналом застосунку є здатність користувача переглядати новини, розслідування, і щотижневі дайджести і статистику.

Діаграма варіантів використання з ключовими акторами та основними функціями показана на рисунку 2.1:

A diagram of a person with text

AI-generated content may be incorrect.

Рисунок 2.1 – Діаграма варіантів використання

В таблицях 2.1-2.13 наведено опис варіантів використання програмного забезпечення.

Таблиця 2.1 – Варіант використання UC-01

| Use case name | Перехід на сторінку новин |
| --- | --- |
| Use case ID | UC-01 |
| Goals | Користувач бачить список статей новин |
| Actors | Користувач |
| Trigger | Користувач натискає на кнопку «Новини» |
| Pre-conditions | - |
| Flow of Events | Користувач заходить на сторінку зі списком новин. Система відображає список статей, які позначені заголовком, виданням, та часом публікації. Статті відсортовані у порядку спадання за часом публікації. Система також надає можливість фільтрувати статті. |
| Extension | - |
| Post-Condition | Користувач знаходиться на сторінці новин |

Таблиця 2.2 – Варіант використання UC-02

| Use case name | Застосування фільтру |
| --- | --- |
| Use case ID | UC-02 |
| Goals | Список статей обмежується за параметрами, виставленими користувачем |
| Actors | Користувач |
| Trigger | Користувач натискає на позначку фільтра на сторінці новин |
| Pre-conditions | Користувач знаходиться на сторінці новин |
| Flow of Events | Користувач натискає на позначку фільтра на сторінці новин. Відкривається вікно фільтра. Користувач може закрити вікно фільтра, виставити параметри пошуку за замовченням або зберегти налаштування. |
| Extension | - |
| Post-Condition | Користувач, після налаштування фільтра, повертається на сторінку новин |

Таблиця 2.3 – Варіант використання UC-03

| Use case name | Обмежити пошук статей по виданню |
| --- | --- |
| Use case ID | UC-03 |
| Goals | Користувач може бачити на сторінці новин статті тільки від тих видань, від яких користувач бажає |
| Actors | Користувач |
| Trigger | Користувач натискає на позначку фільтра на сторінці новин |
| Pre-conditions | Користувач знаходиться на сторінці фільтра |
| Flow of Events | Знаходячись на сторінці фільтра, користувач може обирати за допомогою check-box видання, від яких він бажає бачити статті. |
| Extension | - |
| Post-Condition | Користувач залишається на сторінці фільтра. |

Таблиця 2.4 – Варіант використання UC-04

| Use case name | Обмежити пошук статей по оцінкам якості |
| --- | --- |
| Use case ID | UC-04 |
| Goals | Користувач бачить на сторінці новин тільки ті новини, які підпадають по параметрам якості виставленим користувачем у фільтрі |
| Actors | Користувач |
| Trigger | Користувач натискає на позначку фільтра на сторінці новин |
| Pre-conditions | Користувач знаходиться на сторінці фільтра |
| Flow of Events | Користувач, у сторінці фільтра може використати такі елементи як check-box та range slider для обмеження статей по метрикам емоційності, сенсаційності, фактичності та достовірності. |
| Extension | - |
| Post-Condition | Користувач залишається на сторінці фільтра. |

Таблиця 2.5 – Варіант використання UC-05

| Use case name | Переглянути сторінку статті |
| --- | --- |
| Use case ID | UC-05 |
| Goals | Користувач знаходиться на сторінці статті |
| Actors | Користувач |
| Trigger | Користувач натискає на заголовок статті на сторінці списку статей |
| Pre-conditions | Користувач знаходиться на сторінці списку статей |
| Flow of Events | Користувач, по натисканню заголовку статті, переходить на сторінку статті. На цій сторінці користувач може переглянути заголовок статті, текст статті, джерела, та посилання на сторінку статті від видання, з якого воно було взяте. |
| Extension | - |
| Post-Condition | Користувач залишається на сторінці статті. |

Таблиця 2.6 – Варіант використання UC-06

| Use case name | Переглянути метрики якості |
| --- | --- |
| Use case ID | UC-06 |
| Goals | Користувач може побачити оцінки статті |
| Actors | Користувач |
| Trigger | Користувач натискає на заголовок статті на сторінці списку статей |
| Pre-conditions | Користувач знаходиться на сторінці списку статей |
| Flow of Events | Нижче заголовку і тексту статті, користувач може побачити метрики якості, які були присвоєні статті. На сторінці це відображається у форматі метрика – оцінка. При наведені курсору на метрику, користувач може побачити пояснення від ШІ, чому саме така оцінка була поставлена. |
| Extension | - |
| Post-Condition | Користувач залишається на сторінці статті. |

Таблиця 2.7 – Варіант використання UC-07

| Use case name | Переглянути схожі новинні статті |
| --- | --- |
| Use case ID | UC-07 |
| Goals | Користувач бачить схожі до статті, на сторінці якої користувач зараз перебуває, статті |
| Actors | Користувач |
| Trigger | - |
| Pre-conditions | Користувач знаходиться на сторінці статті |
| Flow of Events | Під метриками якості на сторінці статті, користувач може побачити схожі статті. Ці статті подаються у форматі заголовку, видання, що опублікувало статтю і часу публікації. По натисканню на схожу статтю, користувач може моментально потрапити на сторінку статті. |
| Extension | - |
| Post-Condition | Користувач знаходиться на сторінці схожої статті. |

Таблиця 2.8 – Варіант використання UC-08

| Use case name | Переглянути сторінку розслідувань факт-чекерів |
| --- | --- |
| Use case ID | UC-08 |
| Goals | Користувач знаходиться на сторінці зі списком розслідувань від факт-чекерів |
| Actors | Користувач |
| Trigger | Користувач натискає на кнопку «Викривання фейків» |
| Pre-conditions | - |
| Flow of Events | Користувач потрапляє до списку розслідувань. Розслідування, знаходяться в такому самому форматі як і статті новин. |
| Extension |  |
| Post-Condition | Користувач знаходиться на сторінці з розслідуваннями. |

Таблиця 2.9 – Варіант використання UC-09

| Use case name | Переглянути сторінку розслідування |
| --- | --- |
| Use case ID | UC-09 |
| Goals | Користувач бачить дані про розслідування |
| Actors | Користувач |
| Trigger | Користувач натискає на заголовок розслідування |
| Pre-conditions | - |
| Flow of Events | Користувач переходить на сторінку розслідування, де він може побачити заголовок, посилання на оригінальну публікацію, текст, джерела. |
| Extension |  |
| Post-Condition | Користувач перебуває на сторінці розслідування. |

Таблиця 2.10 – Варіант використання UC-10

| Use case name | Переглянути категорії, які були присвоєні розслідуванню |
| --- | --- |
| Use case ID | UC-10 |
| Goals | Користувач бачить категорії, які були присвоєні розслідуванню |
| Actors | Гість (незареєстрований користувач) |
| Trigger | - |
| Pre-conditions | Користувач знаходиться на сторінці розслідування |
| Flow of Events | Під заголовком розслідування, користувач може побачити категорії, які були присвоєні розслідуванню під час аналізу LLM. |
| Extension | - |
| Post-Condition | Користувач перебуває на сторінці розслідування |

Таблиця 2.11 – Варіант використання UC-11

| Use case name | Переглянути сторінку щотижневих трендів і дайджесту |
| --- | --- |
| Use case ID | UC-11 |
| Goals | Користувач знаходиться на сторінці щотижневих трендів і дайджесту |
| Actors | Користувач |
| Trigger | Користувач натискає на кнопку «Тренди дезінформації» |
| Pre-conditions | - |
| Flow of Events | Користувач переходить до сторінки з щотижневими трендами і дайджестом. |
| Extension | - |
| Post-Condition | Користувач перебуває на сторінці «Тренди дезінформації» |

Таблиця 2.12 – Варіант використання UC-12

| Use case name | Переглянути згенерований дайджест за тиждень |
| --- | --- |
| Use case ID | UC-12 |
| Goals | Користувач бачить дайджест за останній тиждень у |
| Actors | Користувач |
| Trigger | - |
| Pre-conditions | Користувач перебуває на сторінці «Тренди дезінформації» |
| Flow of Events | Користувач бачить дайджест за останній тиждень. Цей дайджест складається з заголовку «Дайджест за {Дата}», та тексту, який коротко описує об'єкти розслідувань за останній тиждень. |
| Extension |  |
| Post-Condition | Користувач перебуває на сторінці «Тренди дезінформації» |

Таблиця 2.13 – Варіант використання UC-13

| Use case name | Переглянути найпопулярніші категорії дезінформації впродовж часу |
| --- | --- |
| Use case ID | UC-13 |
| Goals | Користувач |
| Actors | Користувач бачить на графіку |
| Trigger | - |
| Pre-conditions | Користувач перебуває на сторінці «Тренди дезінформації» |
| Flow of Events | Користувач бачить статистику найчастіше з категоріями, що найчастіше зустрічаються у розслідуваннях факт-чекерів за останні 3-5 тижнів. Кожна категорія вираховується як сума усіх розслідувань з відповідною категорією. |
| Extension | - |
| Post-Condition | Користувач перебуває на сторінці «Тренди дезінформації» |

## Розроблення функціональних вимог

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. В таблиці 2.14 наведено загальну модель вимог, а в таблиці 2.15 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити на рисунку 2.2.

Таблиця 2.14 – Загальна модель вимог

| № | Назва | ID Вимоги | Пріоритети | Ризики |
| --- | --- | --- | --- | --- |
| 1 | Перехід на сторінку новин | FR-1 | Високий | Низький |
| 2 | Застосування фільтру для новин | FR-2 | Середній | Низький |
| 2.1 | Обмежити пошук статей по виданню | FR-3 | Середній | Низький |
| 2.2 | Обмежити пошук статей по метрикам якості | FR-4 | Середній | Середній |
| 3 | Переглянути сторінку статті | FR-5 | Високий | Низький |
| 3.1 | Переглянути схожі статті | FR-6 | Середній | Низький |
| 3.2 | Переглянути оцінки якості статті | FR-7 | Низький | Низький |
| 4 | Переглянути сторінку розслідувань факт-чекерів | FR-8 | Високий | Низький |
| 4.1 | Переглянути сторінку розслідування | FR-9 | Високий | Низький |
| 4.2 | Переглянути категорії, які були присвоєні розслідуванню | FR-10 | Середній | Низький |
| 5 | Переглянути сторінку щотижневих трендів і дайджесту | FR-11 | Середній | Низький |
| 5.1 | Переглянути згенерований дайджест за тиждень | FR-12 | Середній | Низький |
| 5.2 | Переглянути найпопулярніші категорії дезінформації впродовж часу | FR-13 | Низький | Низький |

Таблиця 2.15 – Перелік функціональних вимог

| Назва | Опис |
| --- | --- |
| FR-1 | Перехід на сторінку новин  Функція дозволяє користувачу переходити на головну сторінку зі списком усіх останніх новинних статей від підтримуваних видань. Система відображає статті у хронологічному порядку дати публікації статей з можливістю навігації. |

Продовження таблиці 2.15

| Назва | Опис |
| --- | --- |
| FR-2 | Застосування фільтру для новин  Система надає функціонал фільтрації новинного контенту за різними критеріями. Користувач може застосувати комбінацію фільтрів для знаходження статей що відповідають його інтересам. |
| FR-3 | Обмежити пошук статей по виданню  Користувач має можливість фільтрувати статті за конкретними новинними видавниками (Espreso, Hromadske, Ukrinform, Радіо Свобода). Система дозволяє обирати один або кілька видавників одночасно. |
| FR-4 | Обмежити пошук статей по метрикам якості  Функція дозволяє користувачу фільтрувати статті за показниками якості: емоційність, сенсаційність, фактичність, достовірність джерел. Кожна метрика має свою шкалу оцінювання. |
| FR-5 | Переглянути сторінку статті  При натисканні на заголовок статті користувач переходить на детальну сторінку з повним текстом статті, видавником, часом публікації та посиланням на оригінальну публікацію. |
| FR-6 | Переглянути схожі статті  На сторінці статті система відображає список семантично схожих статей, знайдених за допомогою семантичного аналізу векторів статей. Користувач може перейти до будь-якої зі схожих статей. |
| FR-7 | Переглянути оцінки якості статті  Система відображає автоматично згенеровані оцінки статті за метриками: емоційність, фактичність, достовірність джерел, сенсаційність. При наведенні на оцінку відображається пояснення від AI. |

Продовження таблиці 2.15:

| Назва | Опис |
| --- | --- |
| FR-8 | Переглянути сторінку розслідувань факт-чекерів  Користувач має доступ до окремої сторінки із списком розслідувань дезінформації від організацій VoxUkraine та StopFake. Статті відображаються в хронологічному порядку. |
| FR-9 | Переглянути сторінку розслідування  Детальний перегляд факт-чекінгового розслідування з повним текстом, джерелами, та посиланням на оригінальну публікацію. Відображається контекст викритого фейку. |
| FR-10 | Переглянути категорії, які були присвоєні розслідуванню  Система автоматично категоризує розслідування за типами дезінформації (виправдання агресії, маніпуляції з військовими діями, політичні процеси тощо). Категорії відображаються як теги. |
| FR-11 | Переглянути сторінку щотижневих трендів і дайджесту  Користувач має доступ до аналітичної сторінки з оглядом тенденцій дезінформації. Сторінка містить графіки трендів та резюме подій за тиждень. |
| FR-12 | Переглянути згенерований дайджест за тиждень  Система автоматично генерує короткий текстовий огляд основних дезінформаційних кампаній за попередній тиждень, створений за допомогою LLM на основі зібраних розслідувань. |
| FR-13 | Переглянути найпопулярніші категорії дезінформації впродовж часу  Відображення графіків та статистики, які показують динаміку поширеності різних типів дезінформації протягом останніх тижнів. Дані представлені у вигляді інтерактивних діаграм. |

A grid of lines with dots

AI-generated content may be incorrect.

Рисунок 2.2 – Матриця трасування вимог

## Розроблення нефункціональних вимог

1. Система має бути швидкодійною;
   1. Завантаження сторінок не має займати більше трьох секунд
   2. Час на аналіз і веб-скрейпінг статей має займати стільки часу, щоб не перевантажувати систему
2. Система має бути зручною у використанні;
3. Система має бути захищена від:
   1. CSRF-атак,
   2. XSS-атак,
   3. SQL-ін’єкцій
4. Система має бути сумісною з різними браузерами

## Аналіз системних вимог

Системні вимоги не висовуються.

## Аналіз економічних показників програмного забезпечення

Для аналізу економічних показників ПЗ я прийняв рішення обрати базову модель COCOMO.

Припустимо, що проект має 3,000 вихідних рядків коду, отже проект належить до типу «Розповсюджений» (<50K SLOC)

Відповідно до рисунку 2.4, приймаємо параметри

A table with numbers and symbols

AI-generated content may be incorrect.

Рис. 2.3 – Таблиця коефіцієнтів моделі COCOMO

Отже, можемо визначити економічні показники:

Показник PM – означає трудомісткість, а показник TM – означає орієнтований час на розробку у календарних місяцях.

Отже, при TM = 5.36, та середній заробітній платні у 1800 $ розробника відповідно до DOU.ua [17], витрати на створення дане ПЗ становлять:

EXP = 3300 \* 5.36 = 17,688 $

2.6 Постановка завдання на розробку програмного забезпечення

Отже, розробка програмного забезпечення полягатиме у наступних ключових етапах:

1. Побудова програмного забезпечення для автоматичного веб-скрейпінгу статей від різних видань та розслідувань від факт-чекерів
2. Створення алгоритму для аналізу новинних статей та розслідувань
3. Інтеграція сервісів LLM та алгоритмів для векторизації тексту
4. Налаштування традиційної бази даних і векторної бази даних
5. Створення веб-сторінок для забезпечення взаємодії користувача і сервісу
6. Написання запитів до баз даних для показу актуальної інформації

## Висновки до розділу

Отже, у цьому розділі було детально проаналізовано вимоги до програмного забезпечення.

У першому підрозділі було створено діаграму UML Use-Case для візуалізації варіантів використання застосунку. Також було описано кожен варіант використання. Всього було наведено 13 варіантів використання.

У другому підрозділі було створено список функціональних вимог, яких теж виявилось 13, було детально описано кожну вимогу. Також у цьому підрозділі було створено таблицю трасування вимог і варіантів використання.

У третьому підрозділі було зазначено перелік нефункціональних вимог, де було зазначено вимоги до швидкодії, безпеки, зручності та сумісності.

У підрозділі про аналіз економічних показників було взято метод базової моделі COCOMO та підраховано показники трудомісткості, часових вимог і потенційних витрат.

В кінці було виконано узагальнену постановку задачі, де було наведено ключові етапи в розробці програмного забезпчення.

За результатами розділу сформовано технічне завдання на розробку програмного забезпечення.

# КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Архітектура програмного забезпечення

Як було зазначено у вимогах до програмного забезпечення, застосунок повинен виконувати велику кількість задач, деякі з яких взаємодіють одна з іншим. У проектуванні і конструюванні програмного забезпечення, найголовнішими підходами до архітектури є монолітна і мікросервісна архітектури.

Монолітна архітектура передбачає створення програмного забезпечення всередині одного компонента, де всі модулі тісно переплетені і розгортаються разом. Перевагами монолітної архітектури програмного забезпечення є простота розробки, зручність тестування, і легке розгортання. Проте дуже важливими недоліками є складність масштабування, та високий ризик загального збою усього компонента при збої навіть в одному з модулів[18].

Тим часом, мікросервісна архітектура розбиває застосунок на набір невеликих автономних компонентів, які відповідають за відповідну функцію. Дані сервіси взаємодіють через свій власний API[19]. Перевагами даного підходу до розробки є висока гнучкість і незалежне масштабування, та ізоляція відмов, які компенсуються недоліками у виді складністю управління, вищої затримки та складності тестування.

Для розробки свого програмного забезпечення, було обрано гібридний підхід, який дозволяє об’єднати переваги мікросервісної архітектури у плані стійкості до відмов та гнучкості, та переваги монолітної архітектури, у виді простоти розгортання і розробки.

Для візуалізації архітектури програмного забезпечення, було використано діаграми моделі C4[20]. Діаграма найвищого рівня моделі – це контекстна діаграма, яка дає змогу продемонструвати загальну картину, представляє взаємодію акторів і систем. Діаграму для ПЗ наведено на рисунку 3.1:

A diagram of a software system

AI-generated content may be incorrect.

Рис. 3.1 – Контекстна діаграма

Наступний рівень моделі С4 – це діаграма контейнерів, яка більш детально описує компоненти системи, продемонстрована на рисунку 3.2.

A diagram of a company

AI-generated content may be incorrect.

Рис. 3.2 – Діаграма контейнерів

Діаграма зображує зв’язки наступних елементів:

1) Користувач (Актор);

2) Web-Application (Laravel) – веб-застосунок який надає інтерфейс для доступу до проаналізованих статей і розслідувань;

3) База даних (Postgres) – База даних яка використовується для збереження усіх результатів аналізу;

4) Векторна база даних (Milvus) – Колекція для зберігання векторів статей;

5) Сервіс для аналізу новин (Python) – Функція для отримання статей, виконання аналізу за допомогою LLM, та потім виконання семантичного аналізу за допомогою векторної бази даних Milvus. Результати аналізу зберігаються у базі даних Postgres;

6) Сервіс для аналізу розслідувань (Python) – Функція для отримання розслідувань від факт-чекерів, та надання певних категорій методом аналізу за допомогою LLM.

7) Сервіс для оновлення рекомендацій (Python) – Функція для оновлення схожих статей для усіх новинних статей, час публікації яких менше або дорівнює годині. Використовує векторну базу даних для пошуку найближчого вектора та базу даних для збереження результатів;

8) Сервіс для щотижневих дайджестів (Python) – Функція для створення статистики розслідувань факт-чекерів за останній тиждень, та написання дайджесту на основі даних розслідувань;

9) Планувальник завдань (Celery) – Система для планування завдань, що використовує чергу завдань Redis для виконання сервісу для аналізу новин кожні 15 хвилин; сервісу для оновлення рекомендацій кожну годину; сервісу для збору і аналізу розслідувань кожні 24 години; та виконання сервісу для щотижневого аналізу розслідувань виконуються кожні 7 днів.

Наступний етап – це створення діаграм третього рівня моделі С4, тобто діаграм компонентів. Дані діаграми ще глибше занурюються у архітектуру системи, тому кожен компонент, який на попередній діаграмі не розписувався, матиме свою діаграму.

На рисунках 3.3 – 3.6 зображено діаграми компонентів для вище описаних сервісів 5) - 8).

A diagram of a company

AI-generated content may be incorrect.

Рисунок 3.3 – Діаграма компонента «Сервіс для збору і аналізу новин»

A diagram of a company

AI-generated content may be incorrect.

Рисунок 3.4 – Діаграма компонента «Сервіс для збору і аналізу розслідувань»

A diagram of a diagram

AI-generated content may be incorrect.

Рисунок 3.5 – Діаграма компонента «Сервіс для оновлення схожих статей»

A diagram of a diagram

AI-generated content may be incorrect.

Рисунок 3.6 – Діаграма компонента «Сервіс для створення щотижневого »

## Архітектурні рішення та обґрунтування вибору засобів розробки

Було прийнято рішення використати наступні мови програмування:

Для серверної частини програмного забезпечення було прийнято рішення використати мову програмування Python, через її нескладний синтаксис, широкий перелік бібліотек для аналізу та веб-скрейпінгу, і також через широкий вибір взаємодій з базами даних.

Django надає потужні засоби для швидкої розробки з автоматичною адмін-панеллю та відмінним ORM. Основною перевагою Django є інтеграція з екосистемою Python для машинного навчання та аналізу даних. Однак для даного проєкту Laravel[21] обраний через кращу інтеграцію з фронтенд-рішеннями, простішу конфігурацію хостингу та менші вимоги до серверних ресурсів. Laravel також забезпечує більш зручну роботу через вбудовані засоби.

У контексті веб-застосунку для аналізу новин Laravel надає оптимальний баланс між швидкістю розробки та функціональністю. Eloquent ORM дозволяє ефективно працювати зі складними відношеннями між статтями, метриками та категоріями. Вбудована система міграцій забезпечує версійний контроль структури бази даних, що критично важливо для проєкту з еволюційною схемою даних.

Blade шаблонізатор Laravel спрощує створення динамічних інтерфейсів для відображення новин та їх аналізу. Artisan CLI автоматизує рутинні завдання розробки та дозволяє створювати кастомні команди для специфічних потреб проєкту, таких як запуск скрапінгу або генерація звітів.

Для баз даних, було обрано СУБД PostgreSQL[25] через її швидкодію та сумісність з різними інструментами, та Milvus[24] як векторну базу даних через її сумісність з Python у вигляді бібліотеки pymilvus, яка дозволяє легко і зручно взаємодіяти з колекціями, при локальному розгортанні.

Система не передбачає складної системи аутентифікації користувачів. Всі функції доступні без реєстрації, що спрощує доступ до інформації в умовах швидко змінюваної новинної обстановки. Це рішення обґрунтовується публічним характером новинного контенту, необхідністю швидкого доступу до інформації та спрощенням архітектури системи.

Система також інтегрується з наступними зовнішніми сервісами:

1. OpenAI API[22]: використовується для аналізу якісних характеристик статей (емоційність, фактичність, достовірність джерел, сенсаційність). Нещодавня модель GPT-4.1-Mini обрана завдяки оптимальному співвідношенню якості аналізу, вартості використання, та швидкодії.
2. VoyageAI API[23]: створює семантичні вектори статей для пошуку схожого контенту. Модель voyage-3 забезпечує високу якість векторизації українського тексту.
3. Веб-скрапінг українських медіа: система автоматично збирає новини з провідних українських видань (ukrinform, hromadske, espreso.tv, Радіо Свобода, StopFake, VoxUkraine) за допомогою бібліотек, доступних у Python Selenium[26] та requests[27].

Система відповідає нефункціональним вимогам:

1. Використання СУБД PostgreSQL для відображення матеріалу через об’єктно-реляційну модель Eloquent для Laravel дає змогу задовольнити вимогу у швидкодії.
2. Також використання Eloquent дозволяє запобігти SQL-ін’єкціям.
3. Фреймворк Laravel також є сумісним для браузерів з версіями JavaScript ES6+
4. Зручність забезпечена інтуїтивним інтерфейсом

Для системи черг було обрано Celery через свою здатність ефективно

асинхронно виконувати задачі, які було описано у діаграмах компонентів (рис. 3.3 – 3.6).

Як БД для повідомлень було обрано Redis, що використовує БД в пам’яті для брокерування повідомлень для Celery та кешування часто запитуваних даних.

Для середовищ розробки було обрано PyCharm & PhpStorm від JetBrains, через їх сумісність з версіонуванням, а також великої кількості плагінів, автодоповнення за допомогою JetBrains-AI, та інших невеликих, але зручних переваг.

## Конструювання програмного забезпечення

* + 1. Алгоритм пошуку схожих статей

Як було зазначено на діаграмі компонентів “Сервіс для збору та аналізу новин” та “Сервіс для оновлення схожих статей”, для пошуку схожих статей використовується векторна база даних, яка спеціально оптимізована для математичних операцій з векторами.

Перед операцією пошуку, сервіс VoyageAI дає змогу отримати вектор ебедингів на основі архітектури трансформера. У моєму застосунку, дані вектори отримуються з розмірністю [1024; 1], що дає відкриває змогу порівняти напрямки даних векторів у багатовимірному просторі.

Чим більш співнапрямлений вектор до іншого вектора, тим більше вони подібні. З цієї причини було прийнято рішення використати алгоритм косинусу подібності, що вираховується за наступною формулою:

,

де - це кут відхилення між векторами, , - вектори однакової розмірності.

Дана операція дозволяє отримати легко порівняльну метрику. Чим більш подібні вектори, тим їх косинус подібності буде ближчий до одного, та чим ближче косинус до нуля, тим більша семантична різниця між текстами цих векторів. Це дає змогу відрізнити подібні тексти від неподібних, тому у алгоритмі було обрано поріг для схожих статей у 0.5. Тобто Стаття вважається схожою тоді і тільки тоді, коли косинус подібності більше або дорівнює 0.5.

* + 1. Опис структури бази даних

В якості системи управління базами даних використовується PostgreSQL. База даних серверу призначена для зберігання статей та їхнього аналізу. Опис таблиць бази даних наведено у таблицях 3.1-3.14. Модель бази даних наведена на рисунку 3.2.

Таблиця 3.1 – Опис таблиці article

| Назва поля | Тип даних | Опис |
| --- | --- | --- |
| id | serial | Ідентифікаційний номер статті |
| title | varchar | Оригінальний заголовок статті |
| href | int | Посилання на оригінальну статтю |
| outlet | varchar | Назва видання |
| published\_at | datetime | Час публікації оригінальної статті |
| created\_at | datetime | Час занесення статті у базу даних |
| status | varchar | Статус обробки статті |

Таблиця 3.2 – Опис таблиці paragraph

| Назва поля | Тип даних | Опис |
| --- | --- | --- |
| id | serial | Ідентифікаційний номер параграфу статті |
| paragraph\_text | varchar | Текст параграфу |
| paragraph\_num | int | Номер параграфу як це було у статті |
| article\_id | int | Ключ з посиланням на articles |

Таблиця 3.3 – Опис таблиці source

| Назва поля | Тип даних | Опис |
| --- | --- | --- |
| id | serial | Ідентифікаційний номер посилання на джерело в статті |
| source\_url | varchar | Посилання на джерело взяте зі статті |
| source\_num | int | Номер джерела в порядку зростання зі статті |
| article\_id | int | Ключ з посиланням на articles |

Таблиця 3.4 – Опис таблиці recommended\_article

| Назва поля | Тип даних | Опис |
| --- | --- | --- |
| id | serial | Ідентифікаційний номер схожої статті |
| article\_id | int | Ключ з посиланням на article, який вказує на статтю для якої підбираються схожі статті |
| recommended\_article\_id | int | Ключ з посиланням на article, який вказує на схожу статтю |
| similarity\_score | float | Ключ з посиланням на articles |
| last\_updated | datetime | Час, коли була зроблена рекомендація |

Таблиця 3.5 – Опис таблиці metric

| Назва поля | Тип даних | Опис |
| --- | --- | --- |
| Id | serial | Ідентифікаційний номер метрики оцінювань від LLM |
| article\_id | int | Ключ з посиланням на article, який вказує на статтю для якої зроблені оцінки |
| emotionality\_score | varchar | Оцінка емоційності статті (нейтральна, дещо емоційна, дуже емоційна) |
| emotionality\_reason | varchar | Обґрунтування для оцінки емоційності |
| factuality\_score | int | Оцінка фактуальності статті (чи посилається стаття на реальні події) |
| factuality\_reason | varchar | Обґрунтування для оцінки фактуальності |
| credibility\_score | int | Оцінка достовірності статті (наскільки надійні джерела) |
| credibility\_reason | varchar | Обґрунтування для оцінки достовірності |
| clickbaitness\_score | int | Оцінка сенсаційності статті (наскільки гучний заголовок статті) |
| clickbaitness\_reason | varchar | Обґрунтування для оцінки сенсаційності |

Таблиця 3.6 – Опис таблиці factcheck\_category

| Назва поля | Тип даних | Опис |
| --- | --- | --- |
| id | serial | Ідентифікаційний номер присвоєної до статті категорії |
| article\_id | int | Ключ з посиланням на article, який вказує на статтю для якої підбираються схожі статті |
| category\_id | int | Ключ з посиланням на category, який вказує на ідентифікатор категорії |

Таблиця 3.7 – Опис таблиці category

| Назва поля | Тип даних | Опис |
| --- | --- | --- |
| id | serial | Ідентифікаційний номер категорії |
| name | int | Назва категорії |
| description | int | Опис категорії |

Таблиця 3.8 – Опис таблиці weekly\_stats

| Назва поля | Тип даних | Опис |
| --- | --- | --- |
| id | serial | Ідентифікаційний номер статистики |
| category\_id | int | Ключ з посиланням на ідентифікатор категорії |
| num\_articles | int | Кількість статей, які були знайдені за відповідною категорією за останній тиждень |
| date | date | Останній день тижня, на якому збиралися статті |

Таблиця 3.9 – Опис таблиці weekly\_digest

| Назва поля | Тип даних | Опис |
| --- | --- | --- |
| id | serial | Ідентифікаційний номер дайджесту |
| digest\_date | date | Дата останнього дня тижня, коли був створений дайджест |
| report\_text | varchar | Текст дайджесту |

A diagram of a data flow

AI-generated content may be incorrect.

Рисунок 3.2 – ER діаграма усіх сутностей бази даних

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 3.10.

Таблиця 3.10 – Опис утиліт

| №  п/п | Назва утиліти | Опис застосування |
| --- | --- | --- |
| 1 | PyCharm | Головне середовище розробки програмного забезпечення серверної частини курсової роботи. |
| 2 | PhpStorm | Головне середовище розробки програмного забезпечення клієнтної частини курсової роботи. |
| 3 | Composer | Менеджер залежностей для PHP |
| 4 | Git | Система контролю версій, використовувалася для відстежування змін у коді |
| 5 | pip | Менеджер залежностей для Python |
| 6 | Docker | Застосунок для контейнеризації, який використовувався для розгортання векторної бази данх Milvus |
| 7 | SQLalchemy | Бібліотека для Python, яка надає можливість використовувати ORM. |
| 8 | pymilvus | Бібліотека для взаємодії з колекціями векторної бази даних milvus |
| 9 | Requests | Бібліотека для надсилання запитів у Python. Використовувалася для отримання HTML-коду сторінок новин і розслідувань |

Продовження таблиці 3.10

| №  п/п | Назва утиліти | Опис застосування |
| --- | --- | --- |
| 8 | pymilvus | Бібліотека для взаємодії з колекціями векторної бази даних milvus |
| 9 | Requests | Бібліотека для надсилання запитів у Python. Використовувалася для отримання HTML-коду сторінок новин і розслідувань |
| 10 | Selenium | Бібліотека для автоматизації роботи з браузером у Python. Використовувалася для веб-скрейпінгу для сторінок які використовували JavaScript, що унеможливлювало веб-скрейпінг з requests. |
| 11 | openai | Бібліотека для Python від OpenAI, розроблена для полегшення роботи з сервісами від OpenAI |
| 12 | pydantic | Бібліотека для створення спеціальних структур даних. Використовується для генерації стабільних типізованих відповідей від LLM |

Тексти програмного коду наведені в окремому документі «Текст програми».

## Аналіз безпеки даних

Розроблена система має кілька рівнів захисту від різних типів атак та загроз безпеки. В контексті управління вразливостями система використовує Laravel фреймворк який має вбудовані засоби захисту від SQL ін'єкцій завдяки використанню Eloquent ORM та підготовлених запитів. Всі користувацькі вводи проходять валідацію через контролери Laravel та Pydantic схеми в Python частині що запобігає некоректному введенню даних та потенційним атакам через маніпуляцію параметрів. CSRF токени автоматично генеруються Laravel для всіх форм що захищає від Cross-Site Request Forgery атак. Система використовує HTTPS протокол для шифрування трафіку між клієнтом та сервером що запобігає перехопленню даних під час передачі.

Безпека даних забезпечується кількома способами. API ключі для зовнішніх сервісів OpenAI та VoyageAI зберігаються у змінних середовища через dotenv файли та не потрапляють до репозиторію коду. База даних PostgreSQL має обмежені права доступу та використовує з'єднання через захищені паролі. Персональні дані користувачів не збираються оскільки система не передбачає реєстрації або аутентифікації що знижує ризики пов'язані з витоком особистої інформації. Всі зібрані новини містять лише публічно доступну інформацію з офіційних медіа ресурсів.

В сфері тестування безпеки система проходить перевірку через статичний аналіз коду в IDE PyCharm та PhpStorm які виявляють потенційні вразливості на етапі розробки. Laravel має вбудовані засоби захисту від XSS атак через автоматичне екранування змінних у Blade шаблонах. Python код використовує типізацію через Pydantic що попереджає помилки типів даних та можливі експлойти. Веб скрапінг модулі мають обмеження за часом виконання та обсягом завантажуваних даних що запобігає DoS атакам на цільові ресурси.

Архітектура системи має кілька рівнів захисту. Веб додаток Laravel працює як фронтенд інтерфейс та не має прямого доступу до логіки збору новин. Python сервіси працюють ізольовано та взаємодіють з базою даних через встановлені права доступу. Redis брокер для Celery має обмежений доступ лише до необхідних черг завдань. Milvus векторна база даних працює в локальному режимі що знижує ризики несанкціонованого доступу через мережу.

Система не використовує контейнери Docker у базовій конфігурації але може бути розширена для додаткового рівня ізоляції. Всі зовнішні API викликаються через HTTPS що забезпечує шифрування трафіку. Логування системи ведеться на рівні Laravel та Python що дозволяє відстежувати підозрілу активність та виявляти потенційні атаки. Бекап стратегія включає регулярне копіювання бази даних PostgreSQL для відновлення у випадку втрати даних або компрометації системи.

## Висновки до розділу

Отже, у цьому розділі було детально виконано конструювання і архітектура програмного забезпечення.

У першому підрозділі було детально описано архітектуру програмного забезпечення. Було сконструйовано три рівні моделі C4 та відповідні діаграми контексту, контейнерів і компонентів. Було також обрано загальний патерн до розробки програмного забезпечення. Було сформовано компоненти програмного забезпечення.

У другому підрозділі було детально проаналізовано засоби розробки програмного забезпечення, СУБД. Було описано сторонні сервіси та API, які використовуються створеним програмним забезпеченням. Також було названо методи, як програмне забезпечення виконує поставлені нефункціональні вимоги.

У третьому підрозділі було детально описано базу даних застосунку та відповідні сутності. По результатам було створено діаграму сутностей, яка наочно відображає сутності (таблиці бази даних) та їх взаємозв’язки.

В останньому підрозділі було проаналізовано безпеку даних застосунку. Було продумано декілька можливих типів атак, до яких варто адаптувати застосунок. По результатам аналізу ми переконалися, що застосунок є захищеним від таких атак як SQL-ін’єкції, CSRF та XSS.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Vosoughi, S., Roy, D., & Aral, S. (2018). The spread of true and false news online. Science, 359(6380), 1146-1151. <https://doi.org/10.1126/science.aap9559>
2. Starbird, K., Spiro, E., Edwards, I., et al. (2023). Information warfare during armed conflict: Lessons from Ukraine. Proceedings of the CHI Conference on Human Factors in Computing Systems. <https://doi.org/10.1145/3544548.3581518>
3. Wang, W. Y. (2017). "Liar, Liar Pants on Fire": A new benchmark dataset for fake news detection. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. <https://doi.org/10.18653/v1/P17-2067>
4. VoxUkraine: https://voxukraine.org/category/voks-informue
5. StopFake: <https://www.stopfake.org/uk/category/context-ua/>
6. Espreso.tv - https://espreso.tv/
7. Hromadske - https://hromadske.ua/
8. Ukrinform - https://www.ukrinform.ua/
9. Радіо «Свобода» - <https://www.radiosvoboda.org/>
10. Edelman Trust Barometer : <https://www.edelman.com/trust/2023/trust-barometer>
11. Фейк про «біолабораторії» в Україні: <https://voxukraine.org/fejk-biolaboratoriyi-ssha-v-ukrayini-tse-voyenni-ob-yekty-nato>
12. Фейки про події у Бучі: <https://voxukraine.org/fejk-u-buchi-ukrayinski-vijskovi-obstrilyuvaly-budynky-tyh-hto-brav-dopomogu-vid-rosiyan>
13. Фейки про пошкодження інфраструктури: <https://voxukraine.org/fejk-cherez-poshkodzhennya-energetychnoyi-infrastruktury-v-ukrayini-pochnutsya-spalahy-tyfu-ta-chumy>
14. Google News: <https://news.google.com/home>
15. Ground News: <https://ground.news/>
16. NewsGuard: <https://www.newsguardtech.com/>
17. DOU.ua: <https://jobs.dou.ua/salaries/?period=2024-12&position=Software%20Engineer>
18. What is monolithic architecture? <https://www.ibm.com/think/topics/monolithic-architecture>
19. Microservice Architecture Pattern: <https://microservices.io/patterns/microservices.html>
20. The C4 model for visualizing software architecture: <https://c4model.com/>
21. Laravel: <https://laravel.com/>
22. OpenAI API: <https://platform.openai.com/docs/overview>
23. VoyageAI API: <https://www.voyageai.com/>
24. Postgres: https://www.postgresql.org/
25. Milvus: https://milvus.io/
26. Requests: https://pypi.org/project/requests/
27. Selenium: https://www.selenium.dev/