Міністерство освіти і науки України
Український державний університет науки і технологій

Комп'ютерних технологій і систем
(назва факультету)

Комп'ютерні інформаційні технології
(повна назва кафедри)

Пояснювальна записка
до кваліфікаційної роботи
магістр
(ступінь вищої освіти)

на тему: Контекстне дослідження Web-сайту

за освітньою програмою Інженерія програмного забезпечення
зі спеціальності: 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

Виконав: студент групи: ПЗ2121

(підпис студента) / Данило БОГУЦЬКИЙ /
(Ім'я ПРІЗВИЩЕ)

Керівник: (підпис) / доц. Олександра ГОРБОВА /
(посада, Ім'я ПРІЗВИЩЕ)

Нормоконтролер: (підпис) / доц. Світлана ВОЛКОВА /
(посада, Ім'я ПРІЗВИЩЕ)

Консультанти:
Техніко-економічні розрахунки (підпис) / доц. Микола ГНЕНИЙ /
(назва розділу) (посада, Ім'я ПРІЗВИЩЕ)

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (підпис)

Дніпро – 2022 рік

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Український державний університет науки і технологій

Кафедра Комп'ютерні інформаційні технології

«ДО ЗАХИСТУ»

Завідувач кафедри
___ Вадим ГОРЯЧКІН

«__» _____ 20___р.

ДИПЛОМНА РОБОТА

на здобуття освітнього ступеня «магістр»

Галузь знань **12 Інформаційні технології**

Спеціальність **121 Інженерія програмного забезпечення**

Тема **Контекстне дослідження Web-сайту**

Theme **Contextual inquiry of website**

Керівник дипломної роботи доц._____ Олександра ГОРБОВА

Нормоконтролер доц._____ Світлана ВОЛКОВА

Студент групи ПЗ2121 _____Данило БОГУЦЬКИЙ

Student Danylo BOHUTSKYI

Дніпро – 2022

Український державний університет науки і технологій

Факультет Комп'ютерних технологій і систем кафедра Комп’ютерні інформаційні технології

Спеціальність Інженерія програмного забезпечення

«ЗАТВЕРДЖУЮ»
Завідувач кафедри
_____ Вадим ГОРЯЧКІН
(підпис)
«___»_____2022 р.

ЗАВДАННЯ

до дипломної роботи на здобуття ОС магістр
(освітній ступінь)

студента групи

_____
(номер групи) (ПІБ)

1 Тема дипломної роботи: Контекстне дослідження Web-сайту

_____

затверджена наказом по університету від «18» листопада 2021 р. № 690 ст.

2 Термін подання студентом закінченої роботи_____

3 Вихідні дані до дипломної роботи _____

_____

_____

_____

4 Зміст пояснювальної записки (перелік питань до розробки) аналіз сучасних методів розробки і дослідження веб-сайтів,_____ проектування та ророзка ПЗ_____

_____

_____

5 Перелік демонстраційного матеріалу

_____

_____

_____

_____

_____

_____

6. Консультанти (з назвами розділів):

Розділ	Консультант	Підпис, дата	
		завдання видав	завдання прийняв
Техніко-економічні розрахунки	доц. Гнений М.В.		

КАЛЕНДАРНИЙ ПЛАН

ор.	Назва розділів дипломної роботи	Термін виконання розділів роботи	При-мітка
	Вступ		
	Аналіз сучасного стану дослі-дження проблеми за науковими літературними джерелами		від 70 джерел
	Аналіз сучасного стану програм-но-апаратного забезпечення, яке потребує вдосконалення для вирішення проблем дослідження		
	Постановка задачі, технічне завдання	11.10.22 – 17.10.22	30%
	Техніко-економічні показники		
	Розробка інструментальних за-собів дослідження		
	Виконання досліджень	08.11.22 – 14.11.22	60%
	Оформлення тез доповідей		

	Оформлення статті у фаховий журнал		
0	Оформлення пояснювальної за-писки		
1	Розробка демонстраційних мате-ріалів	29.11.22 – 05.12.22	100 %

Дата видачі завдання «18» листопада 2021 р.

Керівник дипломної роботи _____ О.В. Горбова

(підпис) (ПІБ)

Завдання прийняв до виконання _____ Д.В. Богуцький

(підпис) (ПІБ)

РЕФЕРАТ

Об’єктом дослідження є використання методу контекстних досліджень для оцінювання зручності та практичності використовуваних засобів при проектуванні та створення веб-сайтів.

Предметом дослідження є веб-сайти, що утримують в собі такі елементи дослідження як веб-технології, дизайну, зміст у рамках представленої задачі.

Метою роботи є визначення ефективності методу контекстних досліджень та можливість його практичного використання при досліджені веб-сайтів.

Методи дослідження: урахування контексту, в якому використовується веб-сайт, спільна оцінка веб-сайту розробником та користувачем, оцінка веб-сайту у фокусі зручності користування користувачем.

Результати та їх новизна: дослідження робить внесок у визначення практичної застосовності методу контекстного дослідження для якісної оцінки створюваного продукту та зручності для користувача. Результати дослідження дозволяють зробити висновки щодо можливості практичного використання методу контекстних досліджень для отримання якісної оцінки розробки створюваного продукту.

Пояснювальна записка складається зі вступу, 5 розділів, висновків, бібліографічного списку та 4 додатків:

– у вступі описується сутність розробки, її актуальність. Складається з 3 сторінок;

– у першому розділі проведений аналіз складових сучасних методів розробки і способів дослідження веб-сайтів, розглянуті проблеми аналізу, проведений огляд існуючих методів дослідження та програмних рішень. Складається з 20 сторінок;

– у другому розділі проведений аналіз методу контекстних досліджень, виділені можливі критерії для розробки програмного додатку, наведений опис та описані критерії аналізу читабельності, кількості слів, структурних елементів гіпертекстової сторінки. Складається з 6 сторінок;

– у третьому розділі представлений детальний опис програмного середовища, підходів та мови програмування програмного додатку, опис технологічної

платформи, описані використані бібліотеки, наведені діаграми роботи частин системи програмного додатку, описані її елементи. Складається з 23 сторінок;

– у четвертому розділі описано виконані дослідження на основі спроектованого програмного додатку, представлений аналіз результатів експерименту. Складається з 19 сторінки;

– додатки містять технічне завдання й робочий проект.

Таблиць – 10, рисунків – 52, бібліографія – 70 джерел.

Ключові слова: контекстне дослідження, контекстне, аналіз, веб, сторінка, текст, тег.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

ВСТУП

Актуальність теми.

Висока конкуренція в інтернет-середовищі призводить до необхідності постійно генерувати нові способи вдосконалення контенту. В мережі Інтернет налічується велика кількість інформації щодо інструментів для оцінки та відстеження поведінкових факторів на сайті, але, як правило, відсутні інструменти для визначення якісної оцінки зручності та доречності створюваного програмного продукту, з погляду як можливостей розробника, так і уподобань користувача.

Предмет дослідження.

У зв'язку з стрімким поширенням інформаційних технологій, гостро постає питання зворотного зв’язку розробника та користувача, для створення найбільш актуального функціоналу для потреб користувачів кінцевого продукту, з ефективним використанням людського ресурсу та часу. Метод контекстних досліджень як спосіб спільної взаємодії користувача та розробника, виступає в якості предмета дослідження.

Мета і задачі дослідження.

Задачею дослідження є аналіз практичного використання методу контекстних досліджень для вдосконалення якості створюваного продукту, спрощення взаємодії користувача та розробника та установка точних цілей для розробника стосовно виконуваних задач, які формуються на основі потреб користувача.

Мета контекстних досліджень полягає в:

– удосконаленні збору інформації на стадії створення концепту;

– удосконалення фільтрації зібраної інформації, з урахуванням контексту;

– налаштовуванні спільної взаємодії між можливістю та способу технічної реалізації виконавців відповідно до функціональних вимог користувачів;

– удосконалення ефективності роботи з точки зору витраченого часу на розробку та людського ресурсу.

Методи дослідження.

Одним із складових методу контекстних досліджень є збір інформації методом спостереження за користувацькими діями. Проблема полягає в необхідності отримання детального покрокового опису дій з точки зору користувача. Створений продукт повинен вирішувати реальні завдання та органічно вписатися як в соціальне, так і в фізичне середовище користувача.

Окрім цього, може стати за необхідним формування критеріїв оцінки веб-сайту, відповідно до основних принципів проектування, надання оцінки кількісної характеристики з кінцевими висновками.

Наукова новизна.

Метод контекстних досліджень входить в низку так званих «web usability methods», таких як: Card Sorting, Checlists, Prototyping, Surveys, Questionnaires, Pluralistic Walkthroughs, Self-Reporting Logs, Think Aloud Protocol, Focus Groups, Heuristic Evaluation, Feature Inspection.

Практичне значення.

Метод контекстних досліджень є особливо актуальним для веб-розробки, так як весь програмний продукт, веб-сайт, створюється переважно за бажанням користувачів – їх уподобань щодо інтерфейсу, логіки та поведінки кінцевого продукту, на який орієнтуються розробники. На відміну від інших методів, метод контекстних досліджень дозволяє створити синтез користувацьких уподобань та можливостей розробників.

Апробація результатів дослідження та публікації.

Основні положення магістерської роботи доповідалися та були схвалені на XV міжнародній науково-практичній конференції «Сучасні інформаційні і комунікаційні технології на транспорті, в промисловості та освіті», яка відбулася в Українському державному університеті науки та технологій 16 – 17 грудня 2021 року, див. додаток В.

На основні положень магістерської роботи було створено тези доповіді, котрі були схвалені на XV міжнародній науково-практичній конференції «Сучасні інформаційні і комунікаційні технології на транспорті, в промисловості та освіті», яка відбулася в Українському державному університеті науки та технологій 16 – 17 грудня 2021 року, див. додаток В, а також на XV міжнародній науково-практичній конференції «Інформаційні технології і автоматизація – 2022», яка відбулася в Одеському національному технологічному університеті, Інституті комп'ютерних систем і технологій "Індустрія 4.0" ім.П.Н.Платонова 20 – 21 жовтня 2022 року, див. додаток Г.

1 АНАЛІЗ СУЧАСНИХ МЕТОДІВ РОЗРОБКИ І ДОСЛІДЖЕНЬ ВЕБ-САЙТІВ

1.1 Аналіз сучасних складових веб-розробки

Глобальна мережа Інтернет знаходиться у постійній трансформації: вона постійно розвивається відповідно до потреб користувачів. За станом на 2021 рік, у світі налічується щонайменше 4,66 мільярдів користувачів глобальної мережі Інтернет – більше половини населення планети [1].

За даними компанії «We Are Social», число користувачів веб-ресурсами різко зростає кожен рік. Так, з липня 2018 року щодня близько 900 000 людей вперше виходять у глобальну мережу Інтернет. У порівнянні, в 2018 році – 46,8% населення планети виходили в глобальну мережу [2]. Інфографіку цих даних можна побачити на рис. 1.1.

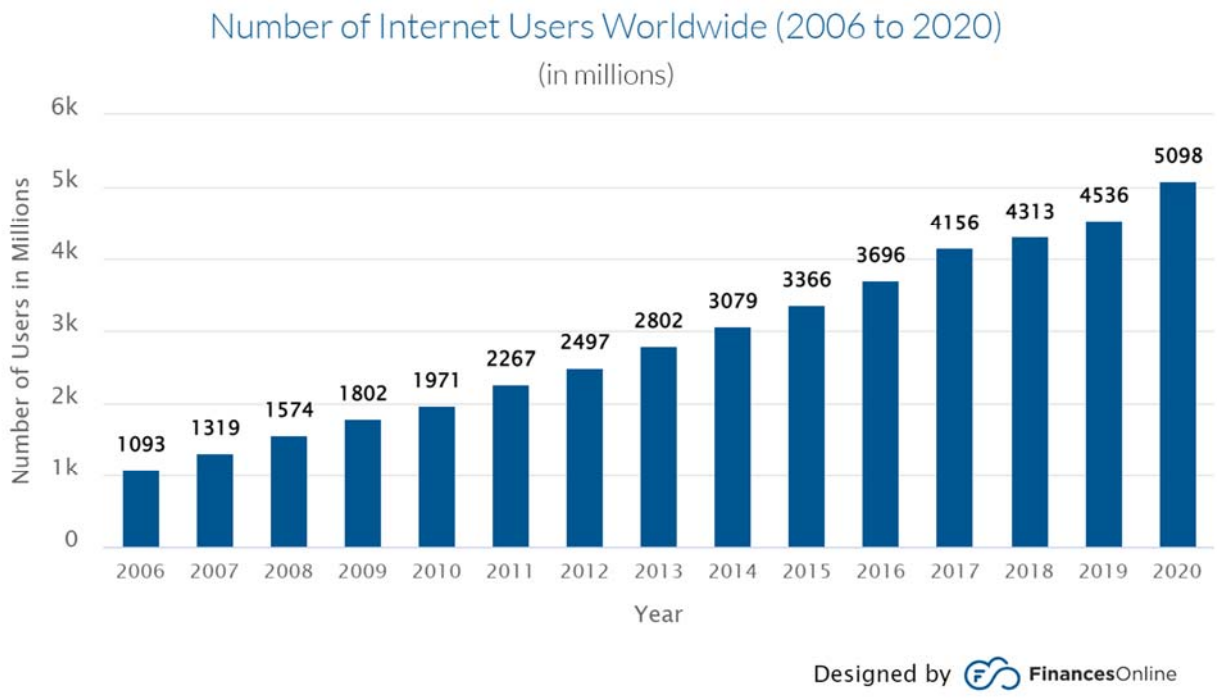


Рисунок 1.1 – Темпи зростання користувачів глобальною мережею Інтернет

Сучасна веб-розробка включає в себе багато складових, насамперед програмних. Стандартний веб-додаток базується на таких елементах:

– база даних;
– веб-сервер;
– операційна система.

Для розробки таких веб-додатків, фірма повинна мати знання та людські ресурси для реалізації цих елементів за допомогою доступних платформ, що пропонує ринок [3].

Кожен програмний елемент виконує свою функцію, для яких він був створений. Так, операційна система є основою, для подальшого встановлення та налаштування програмних продуктів та елементів. На цю програмну платформу, операційну систему, встановлюється веб-сервер, що займається прийняттям, обробкою та передачею HTTP-запитів – вони здійснюється клієнтом до іменованого хосту, який знаходиться на сервері, для доступу до ресурсу на сервері [4]. Користувачі взаємодіють з ними за допомогою спеціальних програм – веб-браузерів. На 2022 рік, найбільш відомими програмними продуктами є такі браузери як Google Chrome, Safari, Firefox, Samsung Internet, Edge, Opera [5]. Рейтинг браузерів та їх долю на ринку на 2022 рік можна побачити на рис. 1.2.

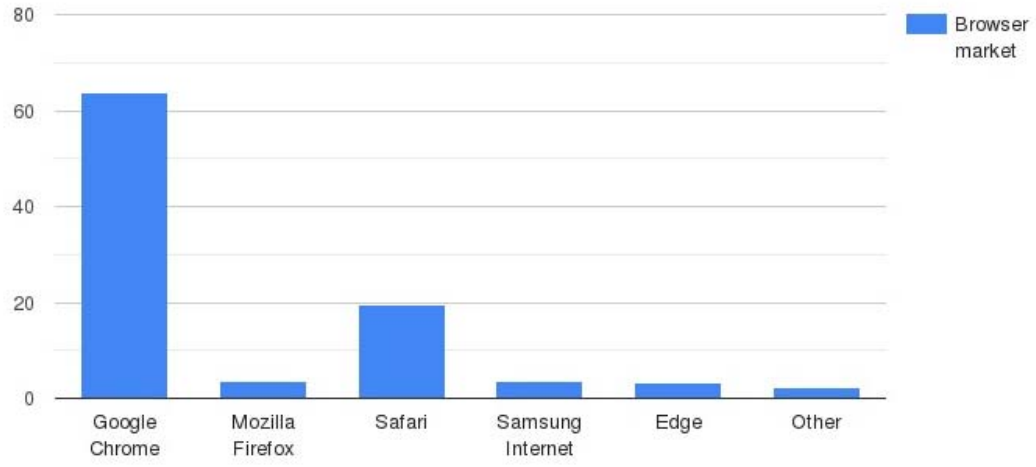


Рисунок 1.2 – Статистика популярності браузерів за 2022 рік

База даних дозволяє зберігати та маніпулювати даними, в залежності від обраної архітектури – реляційні, ієрархічні, об’єктно орієнтовані, сітьові, функціональні і т.д. Дані в найбільш поширених типах баз даних, що

використовуються сьогодні, як правило, моделюються в рядках і стовпчиках в серії таблиць, щоб зробити обробку і пошук даних ефективними [6].

Для розробки веб-додатків використовують спеціалізоване програмне забезпечення – інтегровану середу розробки. Середа розробки поділяється на два види – спеціалізоване та редактор коду. Редактори коду пропонують швидкий та гнучкий інтерфейс, який дозволяє більш ефективно писати код, а також такі можливості, що допомагають аналізувати код на наявність помилок на предмет внесення правок [7].

Однією з головних переваг редактора коду – це те, що це розширені, прості програми. Редактор коду практично не займає місця, мають можливість експлуатуватись у веб-браузер, дозволяючи миттєво приступити до написання коду, що створює велику гнучкість для інструменту [8].

Для опису програмного коду, використовуються різні мови програмування та технології. Так, одна мова призначена для реалізації функціоналу веб-серверу, інша – для клієнтської взаємодії, окрім цього – для роботи з базами даних, налаштування гіпертекстових документів, створення каскадних таблиць стилів і т.д. Так, «Stack Overflow», одна з найбільших платформ для взаємодії програмістів між собою, опитує тисячі розробників про стан їх роботи та актуальний набір технологій, що вони використовують. За даними платформи, за опитом понад , на 2022 рік, найбільш поширенішими мовами веб-програмування є JavaScript (64,96%), HTML/CSS (56,07%), Python (48,24%), SQL (47,08%). В опитуванні приймали участь розробники різних професійних рівнів, більше 80000 осіб [9]. Інфографіка рейтингу мов веб-розробки за опитом платформи надано на рис. 1.3.

Рисунок 1.3 – Рейтинг мов веб-розробки за опитом «Stack Overflow»

1.2 Аналіз сучасної веб-розробки

Сучасна веб-сторінка має відповідати ряду критеріїв, визначених розробником та замовником. При створенні веб-сайту, необхідно враховувати такі аспекти як кінцевий користувач та пошукові системи. Для цього, необхідно розробити таку веб-продукт, з таким набором особливостей, щоб він правильно був індексований пошуковими системами, такими як Google, Bing, Yahoo та зміг бути представлений більшій кількості аудиторії. Для цього розробник має виконувати процес оптимізації такого веб-сайту, який називається SEO. Така оптимізація дозволяє виводити пошуковий результат веб-сайту на більш високі позиції видачі, що стимулює приріст аудиторії [10].

Для визначення критеріїв та виділенням фокус-груп потенційно заінтересованої групи користувачів, до розробки залучаються так звані SEO-спеціалісти. Вони являються фахівцями з маркетингу, використовують дослідження та аналіз для підвищення рейтингу веб-сайту в пошукових системах, таких як Google: вони знаходять найбільш популярні та релевантні ключові слова, що використовуються в запитах пошукових систем, і

вставляють їх на веб-сайти, допомагаючи пошуковим системам знаходити ці сайти та відображати їх користувачам Інтернету [11].

Для визначення критеріїв, ключових слів, функціональних, персональних вимог створюваного продукту, необхідно проводити аналіз потреб користувачів, чим не слід нехтувати – створений продукт навіть за всіма критеріями, визначеними у технічному завданні, не завжди працює ефективно та не є зручним для більшості користувачів. Такі аналізи та дослідження, включають моніторинг потреб та переваг користувачів, конкурентного середовища та поточного становища компанії на ринку, що дозволяє прийняти відповідні рішення з пошуку кращих каналів залучення клієнтів і підвищити ефективність сайту [12].

Окрім роботи з трафіком сайту, сучасна веб-розробка вимагає забезпечувати конфіденційність в глобальній мережі Інтернет. Знання, як захистити свій веб-сайт та убезпечити себе під час роботи в Інтернеті, допоможе зберегти приватну інформацію в таємниці та забезпечити додатковий захист там, де він найбільше потрібен [13].

Для цього необхідно реалізовувати роботу спеціальних протоколів та технологій, таких як SSL сертифікатів. Це цифровий підпис, що гарантує зашифроване з'єднання між веб-сервером та браузером за допомогою протоколу HTTPS. Сертифікат потрібний веб-сайтам, що працюють з особистими даними користувачів, платіжними системами, конфіденційною інформацією.

Існують такі види сертифікатів:

1) SSL-сертифікат з перевіркою домену. Сертифікат підтверджує право на володіння доменом. Перевірку можна пройти за допомогою валідації за почтовою скринькою або за протоколом HTTP. Такі сертифікати можливо отримати як фізичній, так і юридичній особі. В деталях файлу можна побачити центр, що видав сертифікат, термін його дії та алгоритм шифрування.

2) SSL-сертифікат з перевіркою компанії. Сертифікат підтверджує право на володіння доменом, а також що домен належить реально існуючій компанії.

Такий сертифікат видається тільки юридичним особам, а його зміст може відрізнятися, залежно від центру сертифікації, який його видає. Основні дані, що має такий сертифікат – це назва юридичної фірми, свідоцтво про державну реєстрацію бізнесу, приклад документації з назвою організації та контактний номер, що є у відкритих джерелах.

3) SSL-сертифікат із зеленою стрічкою. Аналогічний сертифікату в п.2, однак центр сертифікації проводить більш ретельну перевірку такої організації.

4) SSL-сертифікати з власним підписом. Сертифікат, що не потребує фінансових витрат в центрах сертифікації та може бути виданий самою фірмою, для ідентифікації себе. Недоліком є те, що через відсутність довіреного центру сертифікації, відсутня можливість використовувати захищений протокол HTTPS на веб-сайті, а також браузери повідомляють користувачам, що даний домен не має сертифікату довіри, чим може спричинити відтік трафіку з сайту [14].

1.3 Аналіз проблем веб-розробки

Тенденції веб-розробки стрімко змінюються – на зміну одним технологіям, приходять більш сучасні та ефективніші. В той же час, зростають вимоги до веб-продуктів: більш сучасний інтерфейс, швидкодія, функціональні можливості для проектів і т.д. При створенні проекту, існує ряд факторів, що визначають подальший успіх такого проекту. Замовники хочуть розуміти різні аспекти кінцевого продукту, такі як:

- швидкодія;
- зовнішній вигляд та дизайн;
- співвідношення вартості і якості.

Щоб дізнатися подробиці про команду розробників, клієнти можуть відвідати її «сайт-візитку», мобільні програми та платформи соціальних мереж.

Існують основні проблеми, на які варто зберігати увагу під час розробки будь-якого програмного веб-продукту:

1) Користувацький інтерфейс та досвід. Сучасні веб-додатки мають надавати простий та орієнтований на користувача веб-інтерфейс. При розробці, мають бути створені прототипи сторінки як для звичайних користувачів персональних комп'ютерів, так і для користувачів, що користуються сайтом зі смартфону. Навігація по веб-сайту має бути інтуїтивною та зрозумілою, щоб створити найкращий досвід для користувача, збільшуючи, тим самим, його лояльність. При правильному проектуванні сторінок, користувач може легко та інтуїтивно користуватись продуктом, не звертаючи увагу на потенційних конкурентів замовника. Користувач бачить і взаємодіє з інтерфейсом користувача, а не з базовою внутрішньою архітектурою вашої програми. Тому правильне опрацювання цього елемента вплине на те, наскільки вашим клієнтам сподобається користуватися вашим продуктом і наскільки простий ваш продукт у використанні. Почніть з проектування спочатку інтерфейсу, а потім кодуйте внутрішній движок, який його підтримує, а не створюйте спочатку внутрішній движок, а потім обгортку інтерфейсу [15].

2) Масштабованість. Мається на увазі баланс між серверами, що опрацьовують поточний вхідний графік веб-сайту – при збільшені навантаження, більшій кількості користувачів в єдиний момент часу, необхідно передбачити можливість простого розподілу навантаження на декілька серверів та можливість збільшувати максимальну їх кількість. Розробники мають передбачити програмну можливість розподілу серверної потужності, для регулювання трафіку. Для цього, можна використовувати різні архітектурні методі та стилі, наприклад, SOA – проектування програмного забезпечення, у якому послуги надаються іншим компонентам компонентами програми у вигляді протоколу зв'язку з мережі. Її принципи не залежать від постачальників та інших технологій. У сервіс-орієнтованій архітектурі кілька сервісів взаємодіють один з одним одним із двох способів:

за допомогою передачі даних або за допомогою двох або більше сервісів, що координують будь-яку діяльність [16].

3) Швидкодія. Швидкість сайту або продуктивність сайту означає, наскільки швидко браузер здатний завантажувати повнофункціональні веб-сторінки з даного сайту. На швидкодію можуть вливати різні фактори: не ефективний код, відсутня оптимізація баз даних, некерований ріст даних, скачки трафіку напливу користувачів, неефективний розподіл навантаження, сторонні сервіси і т.д [17].

За даними компанії «Maruti Techlabs», яка є провідним надавачем послуг розробки корпоративних програмних додатків Індії – 47% користувачів очікують, що веб-сторінка буде завантажуватись 2 секунди або меньше, 40% користувачів залишають веб-сайт, якщо сторінка завантажується більше ніж 3 секунди [18]. Інфографіку цих даних можна побачити на рис. 1.4.

Рисунок 1.4 – Аналіз впливу швидкодії на користувачів від «Maruti Techlabs»

В розробку програмного продукту, зокрема веб-сайту, входить тісні взаємодії між клієнтами та командою спеціалістів-розробників. Мета – створення продукту, який буде задовольняти всім потребам клієнта. Проект повинен відповідати бюджету замовника, бути готовим за визначений термін графіком або раніше. Такі накладені обмеження, за для отримання результату, створюють наступні проблеми:

1) Замовник не завжди розуміють, що вони хочуть отримати в кінцевому продукті. Як правило, перед початком робіт, команда розробників та клієнт створюють відповідні специфікації та вимоги до проекту, складають технічне завдання. Етап пов'язаний з отриманням інформації від клієнта та вислуховуванням його потреб. Деякі клієнти можуть бути не впевнені у своїх точних вимогах, тому корисно ставити уточнюючі питання, якщо вони здаються неясними або їхній запропонований напрямок не відповідає розумінню дизайну сайту чи технології [19].

2) Замовник не завжди впевнений у власному бюджеті. Перед початком робіт, команді необхідно підготувати діапазон цінових пропозицій щодо власних послуг для замовника. Якщо клієнт не може визначитись стосовно приблизних обсягів інвестицій в проект – необхідно визначатись з низкою тарифних планів або проводити додаткові консультації. Витративши час на точний облік витрат, пов'язаних з розробкою веб-сайту, ви зможете впевнено встановлювати ціну та визначати бюджет клієнтам, як для малого бізнесу, так і корпорацій [20].

3) Складності під час безпосередньої веб-розробки. Команда розробників має відстежувати сучасні тренди в розробці та використовувати ті технології, які є актуальними на ринку.

4) Складності Інтернет-маркетингу. Необхідно самостійно проводити аналіз аудиторії та робити додаткові заходи для залучення широкої її маси [21].

1.4 Огляд існуючих рішень для вирішення проблем веб-розробки

Як і будь-який інший продукт, створюваний в результаті діяльності, є засобом (інструментом), призначеним для вирішення деякого набору завдань. Вивченням закономірностей вибору людьми найбільш оптимального розв'язання різноманітних завдань займається така наука як теорія прийняття рішень. Вона допомагає розуміти логіку, на підставі вибору професіоналів, користувачів тощо. Вибір призводить до наслідків і зазвичай обговорюється у двох окремих, але різних галузях – гілках. Вони поділяються на нормативну та оптимальну теорії прийняття рішень. При аналізі теорії прийняття рішень аналіз часто складається з того, що є оптимальним рішенням, хто може бути тим, хто приймає оптимальне рішення, і як він може дійти цього рішення. Обговорення того, як люди "мають" приймати рішення у певних сценаріях, також є частиною цього дослідження [22].

Одним із методів вирішення проблем розробки є складання технічне завдання – документу, який є основою розробки та випробування програми чи автоматизованої системи. Основна вимога до документу – його зміст має бути односторонньо зрозумілим і не містити двозначні формулювання, містити детальний опис всіх вимог та побажань замовника до створюваної програми, але при цьому не бути надлишковим за змістом. Документ має містити наступні елементи:

- вступ;
- підстава для розробки;
- призначення розробки;
- вимоги до програмного продукту;
- вимоги до програмної документації;

– техніко-економічні показники;

– стадії та етапи розробки;

– порядок контролю та розробки;

– може містити додатки, якщо такі є в наявності.

Технічне завдання є основним документом для розробників, на який вони орієнтуються під час розробки програмного продукту [23].

Окрім технічного завдання, також розробляється документ специфікацій вимог до програмного забезпечення, або, специфікації. Це структурований набір вимог до програмного забезпечення та його зовнішнім інтерфейсам, при розробці яких необхідно враховувати велику кількість даних. Документ містить загальний розділ, де описується цілі, задачі, визначаються терміни, що використовуються при подальшому життєвому циклі програмного забезпечення, визначається потенційна аудиторія. Окрім цього, документ містить загальний опис продукту, його функціональності. Більшу частину документу займає опис алгоритмів та процесів, складання функціональних вимог, вимог до інтерфейсу тощо. Уніфікована та лаконічна специфікація містить мінімально необхідний обсяг інформації, яка необхідна для отримання чіткого уявлення про загальний вид та функціональні можливості завершеного продукту [23].

Існує 4 основних етапи розробки специфікацій [24]:

1) Попередні дослідження. Проводиться аналіз аналогів, досліджується досвід взаємодії: аналіз аудиторії, визначення основних проблем та завдань, складання бізнес-моделі продукту, оцінка рентабельної суми для його розробки.

2) Формування та аналіз вимог. Розробляється тестове середовище або прототип системи – мінімально працююча програма для ілюстрації алгоритму, основної ідеї.

3) Специфікація вимог. Проводиться збір необхідної інформації: опис прототипу, функціональні, користувальницькі, системні вимоги. Визначається

структура програмного забезпечення, встановлюється взаємодія системних компонентів та алгоритмів.

4) Затвердження вимог. Проводиться перевірка описаних вимог, узгоджуються дискусійні та невизначені пункти, здійснюється кінцеве затвердження достатності та повноти вимог.

1.5 Огляд сучасних методів дослідження веб-сайтів

Існують різні методи та підходи до задачі дослідження веб-сайтів, на предмет створення якісної оцінки за певними критеріями: зручність користування, графічне та стилістичне оформлення елементів, зрозумілість та простота використання інтерфейсу і т.д.

Дослідження ведеться у кількох напрямках:

- дослідження цільової аудиторії;
- дослідження самого продукту;
- дослідження та аналіз конкурентів.

Одним із таких напрямків є так звані «Usability Research», що можна інтерпретувати як «дослідження можливості користування».

Задача досліджень – вивчення взаємодії між потенційною аудиторією та електронними медіа-продуктами, зокрема – веб-сайту. Мета досліджень – визначити способи покращення взаємодії користувача із продуктом, зробити досліджуваний продукт більш простим у використанні та кращим для сприйняття, покращити його цінність користувачем. В ході дослідження, проводиться оцінка інтерфейсу продукту на предмет того, чи є він лаконічним та зрозумілим. Окрім цього, проводиться оцінка побажань користувача – чи задовольняє конкретний продукт потребам користувачів [25].

Існують наступні підходи для аналізу, такі як:

- Card Sorting (карточне сортування);
- Prototyping (макетування);
- Questionnaires (опитування);
- Focus Groups (фокус групи);

– Contextual Inquiry (контекстне дослідження).

Card Sorting – метод дослідження, в якому користувач сортує картки із назвами об’єктів по групам. Назва об’єктів записується на картках та роздаються потенційним респондентам. Їх задача – виконати сортування карток у певній відповідності, за власним представленням структури або ієрархії об’єктів та груп. Існує декілька варіантів такого сортування – відкрите, закрите, зворотнє. У відкритому виді, учасники придумають власні назви для груп об’єктів. У закритому – назва груп надається респондентам заздалегідь. У зворотній – надають завдання та пропонують знайти такий варіант картки, що підходить завданню, з переміщенням по дереву ієрархії від верхнього до нижнього рівня. В кожному з варіантів є користь: у відкритому, респонденти можуть встановити назви архітектурних об’єктів, які можуть бути використані під час розробки веб-сайту; у закритому – для перевірки сприйняття та зрозумілості вже створеної архітектури; у зворотній – для перевірки коректності та зручності запропонованої архітектури та навігації веб-сайту, без урахування зовнішнього вигляду [26].

Prototyping – метод дослідження, головним об’єктом якого є прототип, який може бути представлений як абстрактне уявлення про кінцевий продукт – наприклад, у вигляді схематичного зображення сторінки або малюнок форми, так і щось більш інтерактивне – наприклад, у вигляді попередньої версії продукту, з можливістю взаємодії з окремими його елементами та функціоналом. Мета дослідження – представлення власних ідей на основі прототипу, для збору попередніх відгуків, побажань та аналізу сприйняття потенційних користувачів. Основною перевагою такого підходу є можливість попередньої оцінки майбутнього продукту та збір побажань, які можливо конвертувати в задачі на ранньому етапі розробки, без необхідності витрачати додаткові матеріальні та людські ресурси [27].

Questionnaires – метод дослідження, ідея якого полягає у створенні опитувальника для аналізу потреб та оцінки аудиторії кінцевого продукту. Мета дослідження – отримати уявлення про розумовий процес користувача,

використовуючи питання з відкритим текстом. Питання мають широкий спектр, зокрема – питання, що базуються на виставлені оцінки, балів або надання текстового коментаря. Питання можуть стосуватись різних елементів: оцінка швидкодії відкриття сторінок, дизайн та наповнення контентом сайту і т.д [28].

Focus Groups – метод дослідження, головна мета якого – отримати саме ту інформацію, що неможливо отримати через будь-який аналітичний сервіс веб-аналітики чи за допомогою масових опитувань аудиторії, через низку особливостей: глибинні потреби користувача, особливості діяльності респондента, відношення до оцінюваного продукту і т.д. Такий вид дослідження відноситься до якісних. Суть дослідження полягає в створенні групи респондентів – від 5 до 10 чоловік. Респонденти отримують знання про мету досліджування, знайомляться з іншими учасниками процесу. Окрім респондентів – в групі знаходиться модератор, додатково – помічним модератора, що керують процесом, задаючи питання для подальшого обговорення і аналізу. Мета модератора – слідкувати за переговорним процесом, надаючи можливість кожному із респондентів представити свій погляд на поставлене питання та роботи все для уникнення конфліктних ситуація під час дискусії. Відповіді респондентів фіксуються, можливе залучення аудіо або відео запису. Перевага такого методу полягає в отриманні більш неформальних оцінок користувачів до конкретного питання, виявити способи за якими користувачі вирішують власні завдання – за допомогою оцінюваного продукту чи завдяки альтернативним рішенням [29].

Contextual Inquiry – метод дослідження, мета якого – отримання поглибленого розуміння робочої поведінки та дій користувача. Суть дослідження полягає в спостереженні та інтерв’ю з невеликою вибіркою респондентів. Дослідження проходить з урахуванням контексту – респондент знаходиться у звичному для себе середовищі і не обмежений будь-якими правилами поведінки чи інструкціями, виконує звичні операції. При чому, в дослідженні ураховується контекст того, де знаходиться користувач – вдома,

на робочому місці чи в інших містах. Дослідник спостерігає за діями користувача і в процесі виконання тих чи інших дій респондентом, може здійснювати запит до нього, з метою отримання інформації, як і чому респондент виконує ті чи інші дії. Перевагою такого дослідження є можливість поглибленого вивчення поведінки користувача та збір даних його дій, з можливістю їх уточнення, які можна аналізувати для подальшого вдосконалення продукту [30].

1.6 Огляд програмних аналогів

В залежності від обраного методу дослідження, можуть бути використані різні програмні засоби.

Для опитування, найкраще підходять форми. Програмними аналогами таких продуктів можуть виступати:

- Google Form;
- TypeForm;
- JotForm;
- WuFoo.

Перевагою таких сервісів є простий та інтуїтивний інтерфейс, що дозволяє за декілька кроків створити необхідний набір питань, для опиту респондентів. Сервіси є схожими за функціоналом та архітектурним стилем – представляють собою конструктор, із визначеним набором компонентів, які користувач може використовувати для своїх потреб, при створені опитувальника. Оскільки редактор перетягування є простим і зручним у використанні, внесення змін і регулярне оновлення веб-сайту є дуже простим завданням [31].

Сервіси надають можливість зберігати опитувальники та переглядати їх результати, через веб-інтерфейс. Приклад інтерфейсу таких сервісів можна побачити на рис. 1.5.

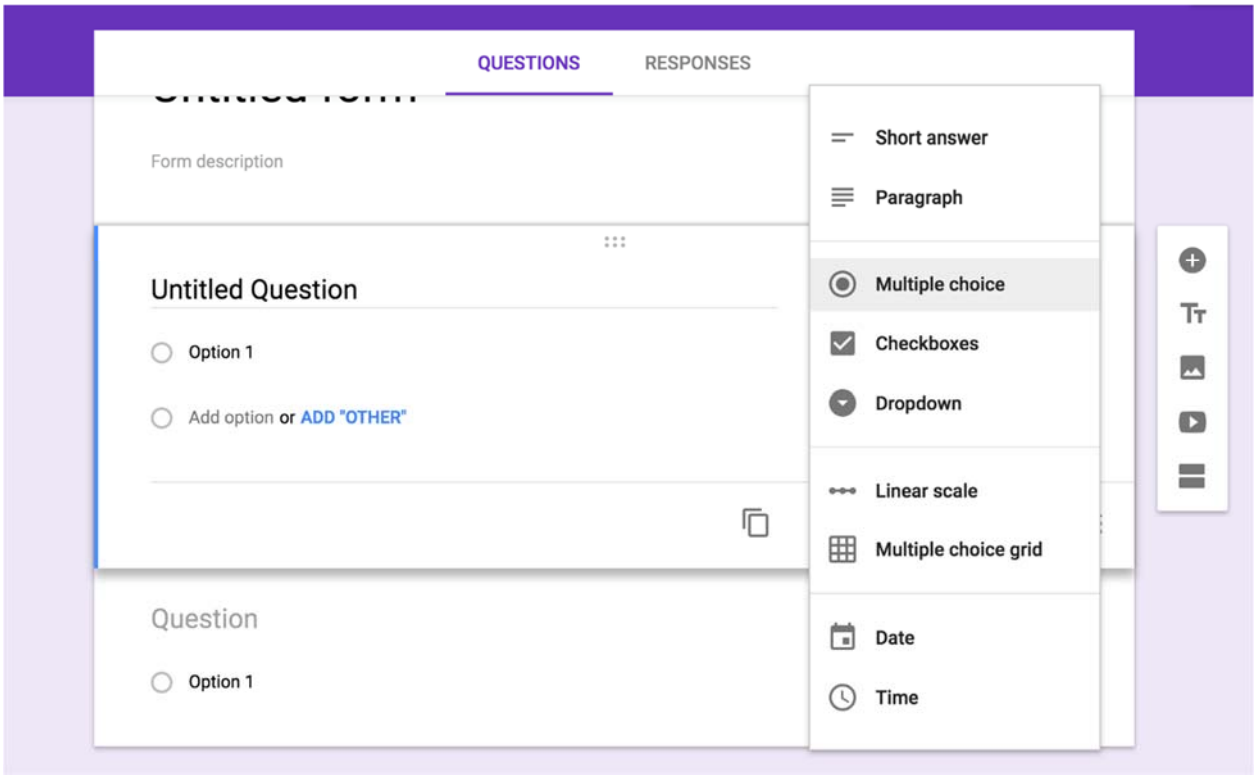


Рисунок 1.5 – Інтерфейс Google Forms

Для сортування картками, існують такі сервіси як:

– OptimalSort;

– UXtweak;

– xSort;

– kardSort.

Вони мають схожі властивості, що і сервіси для створення опитувальників: простий та інтуїтивний інтерфейс, архітектурно та функціонально нагадують конструктор, дають можливість зберігати дані в своїх хмарі, на сервері. Хмарні технології збереження даних робить величезні і постійно зростаючі обсяги цінних даних пов'язаними, доступними і наявними користувачам [32]. Приклад інтерфейсу сервісу для сортування картками наведений на рис. 1.6.

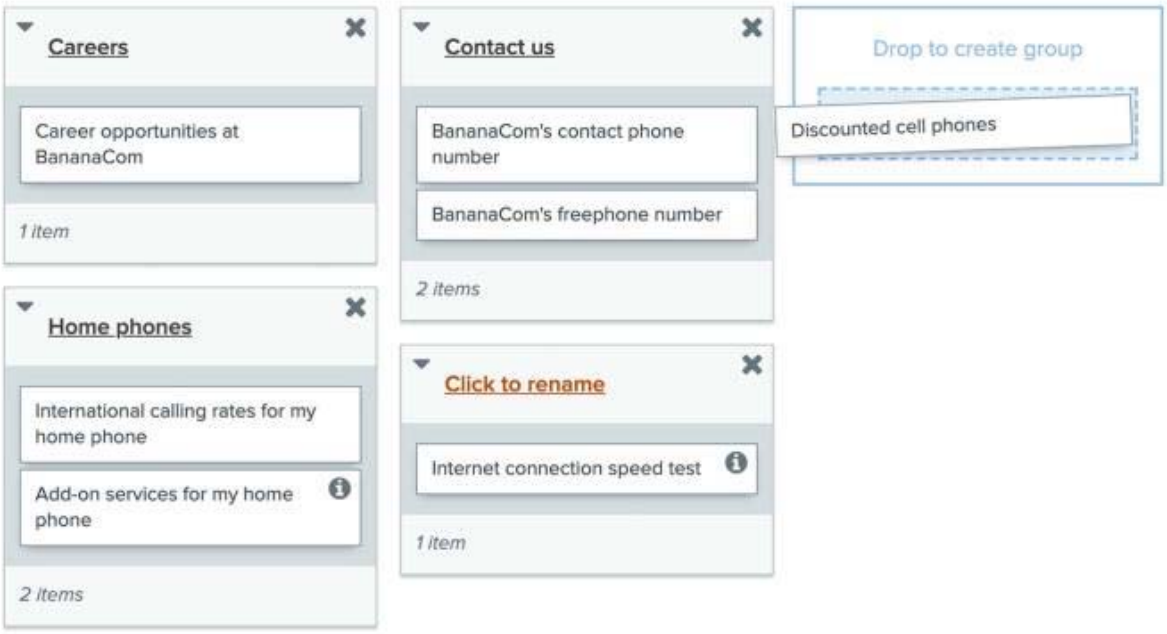


Рисунок 1.6 – Інтерфейс OptimalSort

Для прототипування потребується більш складний функціонал, з можливістю створення макету та його представлення в зручному форматі – локально або за допомогою хмарних технологій, що може надавати сервіс.

Для прототипування, існують такі сервіси як:

– Draw.io;

– Wix;

– proto.io;

– moquaps.

Такі сервіси мають додатковий функціонал, для роботи з файлами, видом перегляду макету, широкою можливістю редагування та персоналізації кожного окремого елементу, можливості до масштабування макету та окремих компонентів. Головна перевага – можливість просто та зручно додавати, редагувати та видаляти об’єкти та переміщувати їх на макеті, із зручними та зрозумілими елементами управління. Приклад інтерфейсу сервісу для прототипування наведений на рис. 1.7.

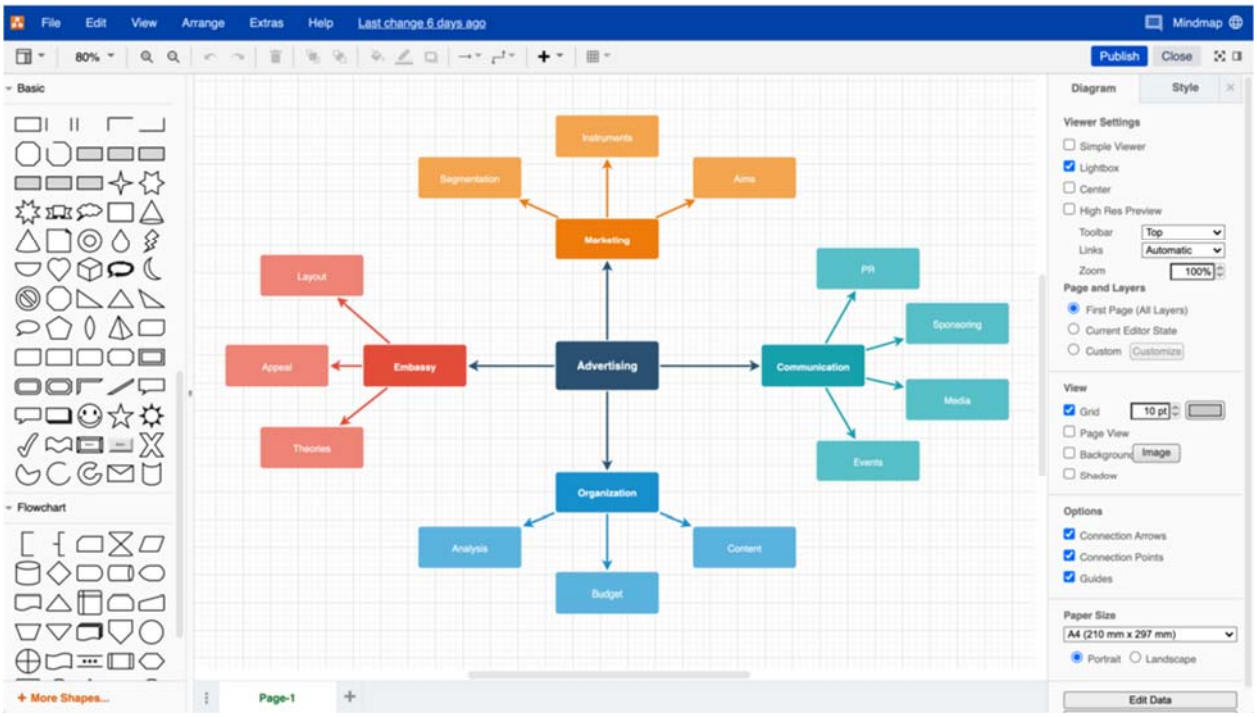


Рисунок 1.7 – Інтерфейс Draw.io

Для фокус-груп, використовуються сервіси комунікації аудіо та відео зв’язку. Вони працюють за допомогою VoIP – протоколом передачі голосу та мультимедійного контенту через інтернет-з'єднання. Він дозволяє користувачам здійснювати голосові дзвінки з різних пристроїв – комп'ютера, смартфона, інших мобільних пристроїв або спеціальних VoIP-телефонів та браузерів із підтримкою WebRTC. Протокол передбачає можливості, такі як: запис розмови, визначення номеру чи голосову пошту на електронну пошту. Вона також корисна для організацій як спосіб уніфікації комунікацій [33].

Для фокус-груп, існують такі сервіси як:

– Skype;

– Discord;

– TeamSpeak;

– Microsoft Teams;

– Zoom;

Приклад інтерфейсу сервісу для фокус-груп наведений на рис. 1.8.

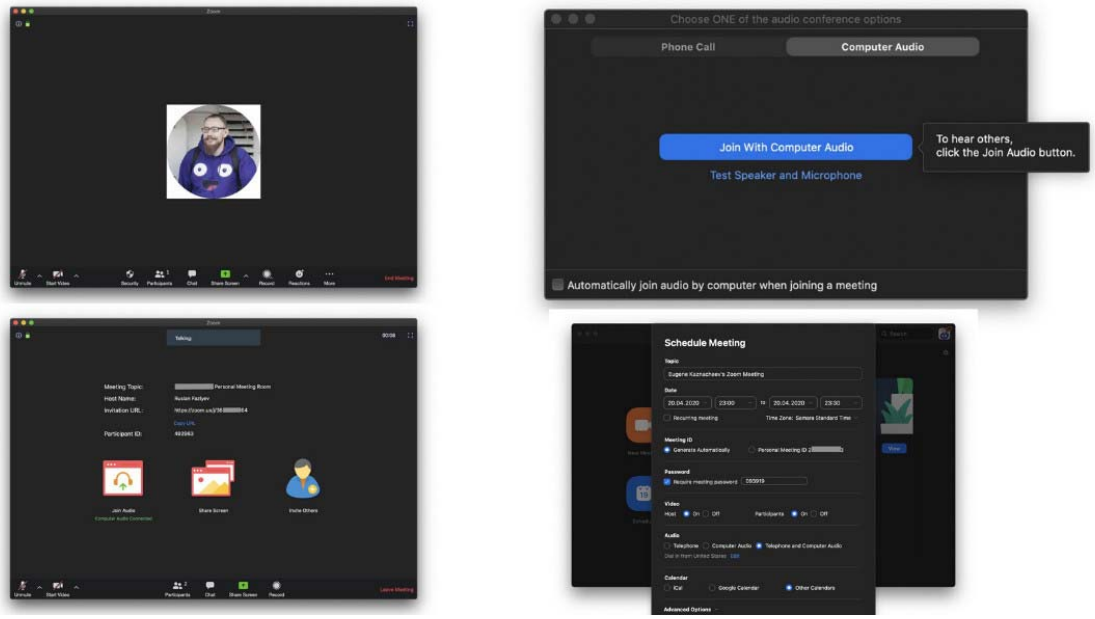


Рисунок 1.8 – Інтерфейс Zoom

Для контекстних досліджень можна частково використовувати такі сервіси як:

– TeamViewer;

– AnyDesk;

– Zoom.

Вони дають можливість переглядати за діями користувача у реальному часі, але не мають функціональних можливостей для зручної комунікації із респондентом для уточнення його дій. Сервіси не ведуть статистичних даних щодо дій користувача, вимушуючи спостерігача власноруч робити помітки та відстежувати всі дії респондента. Окрім описаних факторів – сервіси не є спеціалізованими для аналізу за методом контекстних досліджень, їх призначення – віддалене керування пристроями, на програми встановлені. Тому, їх використання є сумнівним і не дає гарантії ефективного проведення аналізу.

Отже, за наведеною інформацією, для контекстних досліджень, на поточний момент, не існує спеціалізованого програмного забезпечення. Це зумовлено унікальним набором властивостей, яке програмне забезпечення повинно мати та задовольняти функціональним вимогам, які потрібні для

проведення контекстних досліджень та складністю в реалізації такого забезпечення.

1.7 Висновки

При аналізі предметної області, її особливостей та можливих проблем, як для розробника, так і для користувача, з посиланням на результати сторонніх досліджень та їх висновків, можна зробити твердження, що наразі не існує такої програмної реалізації методу, яка може буде повноцінно використовувати метод контекстних досліджень для аналізу веб-сайтів. Через відсутність такого програмного забезпечення, також неможливо провести аналіз доцільності та ефективності такого методу дослідження, так як він може використовуватись для аналізу великої кількості даних, за різними критеріями, в залежності від контексту дослідження, яке проводиться. Наведені програмні альтернативи лише частково можуть бути використані для експериментів, але вони не можуть відображати реальні результати, за якими можна проводити аналіз та робити висновки, так як їх функціональне призначення полягає в інших задачах – тому, вони не здатні дати коректний результат.

На основі аналізу предметної області, необхідно створити формалізацію задачі та критеріїв для програмного додатку, який буде розроблений спеціально для аналізу за методом контекстних досліджень.

2 ОБГРУНТУВАННЯ МЕТОДУ КОНТЕКСТНИХ ДОСЛІДЖЕНЬ ЗА ЕМПІРИЧНОЮ МОДЕЛЛЮ

Перевагою контекстного дослідження являється можливість респондента знаходитись в звичних для себе умовах та виконувати дії, які мають бути конвертовані в спостереження, а результати цих дій мають бути визначені як результати спостережень. Якщо спостереження чи результати потребують уточнення – спостерігач процесу може зробити запит на коментар до респонденту. Недоліками такого підходу є необхідність спостереження за роботою респондента, що виконує певні дії, для досягнення певного результату, визначеного в рамках поставленої задачі або створених умов. Ураховуючи людський фактор в момент спостереження та в цілому аналіз спостережень та їх результатів, різні спостерігачі можуть прийти до різних висновків стосовно досліджених спостережень та на основі їх результату, зробити різні висновки, що може створити дискусію щодо правильності тих чи інших результатів спостережень серед спостерігачів процесу.

Мета дослідження являється проведення серії експериментів з урахуванням методу контекстних досліджень, за допомогою спеціалізованого програмного забезпечення. Особливість цього дослідження полягає в тому, що в процесі експерименту, процес збору та аналізу інформації буде делегована програмному додатку, а не респонденту. Після збору та первинного аналізу даних, за емпіричною моделлю, спостерігач може зробити певні висновки стосовно отриманих результатів.

Програмний додаток оснащений можливістю аналізу за допомогою так званих модулів. Вони являють собою класи, в яких описані алгоритми зборі даних, їх аналіз та формат представлення. Класи можуть реалізовувати інтерфейс, для уникнення проблем з реалізацією методів або являтись потомками абстрактного класу, що задає модель поведінки. Інтерфейс має тільки оголошення методів або абстрактні методи та константні члени даних,

в той час як абстрактний клас може мати абстрактні методи, змінні-члени та конкретні методи [34].

Для проведення серії експериментів, було прийнято рішення відібрати ряд модулів:

1) аналіз тексту сторінки на предмет кількості слів;

2) аналіз тексту сторінки на предмет читабельності;

3) аналіз структурних тегів;

2.1 Аналіз тексту сторінки на предмет кількості слів

Мета будь-якого листа являється створення такого тексту, якому будуть властиві такі риси як читабельність та зручність – то текст, що легко сприймається користувачем. Важливо ефективно висловлювати власні думки, ідеї тощо. Кожне слово має значення: в більшості випадків, краще бути простим та зрозумілим у висловлюваннях, використовувати мінімальну кількість слів. Це робить текст коротшим, приємнішим для читача. Окрім цього, це робить його зрозумілим та легким для сприйняття, повідомлення мають бути прямими і не допускати жодних неправильних тлумачень [35]. Лічильник слів на веб-сторінках є корисним інструментом для SEO-фахівців. За допомогою аналізу слів, вони знаходять сторінку або сторінки з найкращою продуктивністю, можуть оцінити точну довжину для аналізу, щоб використовувати отримані результати для майбутнього створення контенту. Фактори, які зазвичай використовуються для аналізу ключових слів, включають щомісячний обсяг пошуку, рівень конкуренції в пошуковій видачі та потенціал конверсії [36].

Більше того, за допомогою такого інструменту, фахівці не тільки бачать загальну кількість слів, а й густину ключових слів. Це важливо для заповнення веб-сторінок контентом. Ці зміни покращують користувацький досвід та залучають нових відвідувачів [37].

2.2 Аналіз тексту сторінки на предмет читабельності

Читабельність є одним із визначних факторів при просуванні веб-сторінки в глобальній мережі та пошукових системах. Якщо макет перевантажений текстом, а дизайн ускладнює читання – це знецінює будь-які роботи з просуванням веб-сторінки та створює ризики відтоку користувачів. Баланс між інформативністю та лаконічністю, дозволяє усунути ці ризики. Читабельність сайту важлива, оскільки більшість користувачів обмежені в часі та мають обмежені розумові ресурси, які вони можуть витратити на вивчення інформації – вони хочуть мати можливість знайти інформацію, яка їм потрібна, якомога швидше [38].

У міру того як алгоритми пошукових систем обробляють нескінченний потік інформації, вони стають все більш майстерними в розумінні і відтворенні людської мови. Такі програми перевіряють кожне посилання щодо його корисності для читачів, що включає в себе здатність середньої людини зрозуміти повідомлення – ваші читачі матимуть кращий досвід, що допоможе створити почуття довіри та близькості до вашого бренду [39].

Однією з дуже важливих причин збільшення розміру шрифту на сайті є пресбіопія – людина починає відчувати наслідки пресбіопії приблизно у віці близько сорока років. Окуляри для читання стають необхідними, але вони не завжди є, коли вони потрібні [40].

Якщо розмір шрифту занадто малий, а довжина рядка занадто довга, користувачам доведеться напружуватися, щоб прочитати текст, що призведе до фізичних втоми, зокрема очей. Або, якщо розмір шрифту занадто великий (при цьому довжина рядка занадто коротка), користувачі змушені занадто швидко переходити до наступного рядка тексту. Навіть коли односторінковий сайт з довгою прокруткою виглядає неймовірно, необхідно надмірно дбати про ресурс сили та ефективності [41].

За даними цифрового маркетингового агентства KickPoint, 45-80 символів – ідеальна довжина рядка для тексту на веб-сайтах. Остаточна кількість символів у рядку залежить від шрифту, але доки ваш контент вкладається у вказані вище параметри, ви максимально полегшуєте

відвідувачам читання вашого контенту. Для виведення цього правила, агенство дослідило один із найпопулярніших сайтів для публікацій Medium – при максимальній кількості близько 80 символів у рядку (і великому розмірі шрифту в 21px на великих екранах) Medium забезпечує максимальну читабельність та зручність [42].

2.3 Аналіз структурних тегів гіпертекстового документу

Семантичний елемент чітко описує своє значення як браузера, так розробника [43]. Вони не є обов'язковими для функціонування сторінки, але рекомендуються розробникам, для розуміння структури документу. Окрім цього, пошукові системи виводять такі сторінки на більш високі позиції, ніж сторінки, де семантичні теги не вказані, такі як:

- footer;
- header;
- main;
- section;
- nav;
- address;
- і т.д.

Семантичний аналіз, також відомий як семантичне SEO, спрямований на підвищення точності результатів пошуку шляхом розуміння намірів користувача через контекстне значення його пошуку [44].

2.4 Емпіричне дослідження та модель

Основною метою або призначенням дослідження в будь-якій галузі знань є доповнення того, що відомо про досліджуване явище, шляхом застосування наукових методів [45]. Емпіричні дослідження, як правило, спрямовані на пошук загальної історії або пояснення, які застосовуються до різних випадків і в часі. Мета дослідження – створити нові знання про те, як влаштований світ, за якими правилами він працює.

Емпіричне дослідження – це дослідження, висновки якого засновані виключно на конкретних доказах, що піддаються перевірці. Термін "емпіричне" означає, що воно спирається на наукові експерименти та/або докази. Подібним чином дослідження є емпіричним, якщо воно використовує реальні факти при дослідженні своїх тверджень [46].

Емпіричний цикл, який представлено на рисунку 2.1, відображує процес висування гіпотез в тому, як працюють чи поводяться певні суб'єкти, та був перевірки цих гіпотез на емпіричних даних у межах систематичного і суворого підходу. Можна сказати, що він характеризує дедуктивний підхід до науки – починають з соціальної теорії, яка здається переконливою, потім перевіряють її наслідки за допомогою даних – переходять від більш загального рівня до більш конкретного [47]. Схему емпіричного циклу представлено на рис. 2.1.

Рисунок 2.1 – Схема емпіричного циклу

Етапи емпіричного циклу:

1) Спостереження. На цьому етапі виникає ідея про висування гіпотези, збирається емпіричний набір даних.

2) Індукція. Після збору даних, проводиться індукційне міркування для формування загального висновку, на основі першого кроку циклу. На етапі формування гіпотези, для проведення подальшого експерименту для її підтвердження.

3) Дедукція. Виведення висновку з проведеного експерименту. Він має керуватись логікою та раціоналізмом, для отримання правильних результатів.

4) Тестування. Повернення до емпіричних методів для підтвердження гіпотези. Необхідно осмислити результати даних, використати статистичні методи для визначення взаємозв'язку. На цьому етапі формується підтвердження гіпотези, а не її доведення.

5) Оцінка. Проводиться збір даних, аргументи в їх підтримку та висновок на основі проведених експериментів. Вказується обмеження для експерименту та власної гіпотези, рекомендація для більш глибокого дослідження для інших у майбутньому [48].

2.5 Висновки

Метод контекстних досліджень у вигляді програмного додатку повинен мати три модулі або класи – аналіз кількості слів, читабельності тексту, за визначеними критеріями, а також семантичний аналіз тегів. В процесі розробки та експлуатації, при проведені експериментів, на етапі отримання результатів та їх аналізу, можна зробити висновок щодо ефективності такого інструменту та підходу при реалізації модулів контекстного дослідження. На основі цих результатів, можливо буде або зробити корегування вже розроблених модулів, для підвищення їх ефективності або розглянути можливість створення додаткових, для проведення більш детального аналізу сторінок.

3 ПРОЕКТУВАННЯ І РОЗРОБКА ІНСТРУМЕНТАЛЬНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ КОНТЕКСТНОГО ДОСЛІДЖЕННЯ ВЕБ-САЙТУ

3.1 Опис середовища, підходів і мови програмування

В якості середовища для програмування, було обрано інтегровану середу розробки компанії JetBrains PhpStorm 2019.3, що має в собі багатий функціонал для роботи з серверними та клієнтськими мовами програмування. Окрім підтримки різних мов, середа розробки дає можливість працювати з базами даних – переглядати, редагувати, створювати бази дані, таблиці, працювати з колонками тощо. Також, має підтримку модулів, що здатні розширити функціональні можливості середи розробки для подальших потреб, за необхідністю.

PhpStorm був обраний в якості середи розробки через велику функціональність, підтримку багатьох мов програмування та технологій для веб-розробки, можливість розширення функціоналу за допомогою сторонніх модулів та інтеграцією з проектом, індексацію даних, що прискорює та спрощує розробку. PhpStorm забезпечує рефакторинг коду, автодоповнення, запобігання помилок на льоту, налагодження з нульової конфігурації, а також розширений редактор HTML, CSS і JavaScript [49]. Інтерфейс середовища наведений на рис. 3.1.

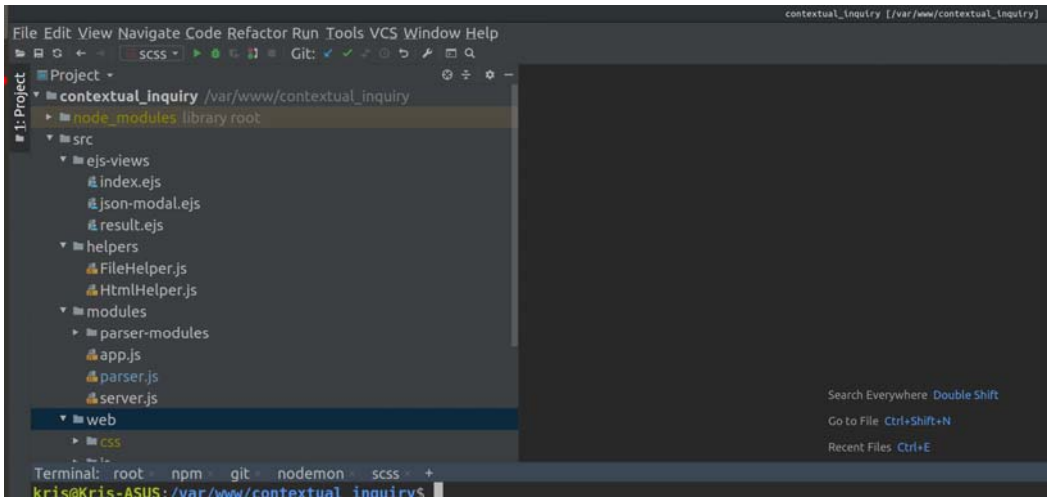


Рисунок 3.1 – Інтерфейс PhpStorm 2019.3

В якості основної мови програмування, було обрано мову JavaScript. Мова програмування JavaScript призначена для роботи на клієнтській частині, для створення інтерактивності на сторінці. Незалежно від того, де розміщений JavaScript, він завжди виконується на клієнтському середовищі, щоб заощадити багато пропускної здатності і зробити процес виконання швидким [50].

Мова програмування JavaScript була обрана в якості основної мови через популярність мови та поширення її підтримки серед всіх сучасних браузерів, простий синтаксис, наявність всіх необхідних засобів та модулів для розробки програмного забезпечення, можливість масштабування за допомогою сторонніх технологій розробки. Гнучкість JavaScript найкраще підходить для розробників середнього рівня. Мова просто допомагає досягти результату, дозволяючи розробнику зосередитися на вирішенні проблеми. Розробники можуть використовувати поєднання плагінів та власних фрагментів коду, щоб змусити програму працювати [51].

Для роботи з стилями, було обрано каскадну таблицю стилів стилів. Для полегшення подальшої розробки та зменшення розмірів створених файлів стилів, було обрано метамова SCSS. Вона слугує для збільшення рівня абстракції при створені таблиць стилів і спрощує процес їх створення та редагування. Використовуючи SCSS дозволить використовувати додаткові функції для CSS, таких як змінні, вкладеність, розширення. Всі ці додаткові функції дозволяють зробити написання стилів набагато простіше і швидше в порівнянні з написанням традиційних стилів [52].

Для контролю версій програмного додатку, а також його збереження, було обрано систему контролю версій Git. В якості платформи для роботи з Git, було обрано сервіс GitHub, на якому буде в подальшому зберігатись проект, містити всю інформацію про нього, зберігати історію змін та модифікацій, можливість завантаження та модифікації. Це платформа для розміщення коду для контролю версій та спільної роботи проекту, яка дозволяє працювати разом над проектами з будь-якого місця [53].

Для початку роботи з GitHub, необхідно завести особисту сторінку розробника, через яку ми будемо здійснювати подальші маніпуляції із проектом. Після створення особистої сторінки, внесення необхідних даних, необхідно створити власний репозиторій, в якому буде зберігатись код проекту, його опис, історія змін та версії. Інтерфейс платформи наведений на рис. 3.2.

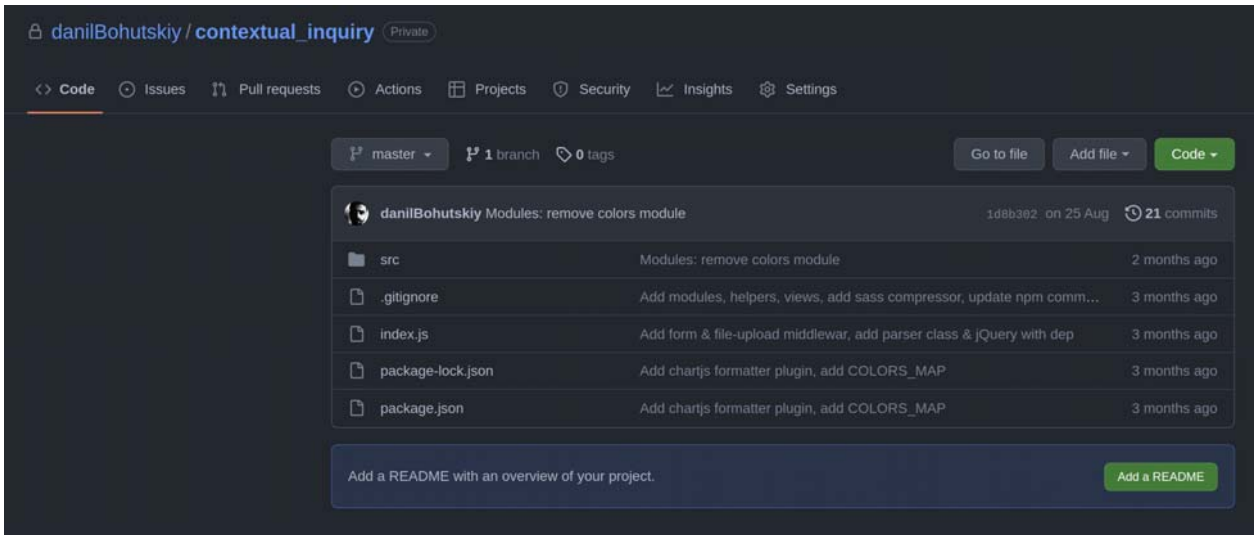


Рисунок 3.2 – Інтерфейс репозиторію проекту на GitHub

Для доступу до проекту, необхідно встановити спеціальні ключі, які будуть гарантувати аутентифікацію та верифікацію розробника, який працює над проектом та має до нього доступ.

SSH називають зашифрований протокол, який використовується для адміністрування та зв’язку із серверами. SSH-ключі забезпечують надзвичайно безпечний спосіб входу на ваш сервер.

Сервер SSH може ідентифікувати клієнтів за допомогою різних методів. Найпростішим з них є аутентифікація за допомогою пароля, який простий у використанні, але такий спосіб найбезпечніший, так як можна підібрати пароль за допомогою спеціалізованого програмного забезпечення.

Не дивлячись на те, що паролі надсилаються на сервер безпечним способом, вони, як правило, не є складними або достатньо довгими, щоб бути стійкими до повторних, наполегливих зловмисників. Сучасні обчислювальні

потужності в поєднанні з автоматизованими скриптами роблять грубий підбір захищеного паролем облікового запису можливим. SSH-ключі виявляються надійною та безпечною альтернативою.

Парою ключів SSH називають два криптографічних захищених ключа, які можуть бути використані для аутентифікації клієнта на SSH-сервері. Кожна пара ключів складається з відкритого і закритого ключів.

Закритий ключ зберігається у клієнта і повинен зберігатися в повній таємниці. Будь-яка компрометація приватного ключа дозволить зловмиснику увійти на сервери, які налаштовані з відповідним публічним ключем, без додаткової аутентифікації. В якості додаткового запобіжного заходу ключ може бути зашифрований на диску за допомогою парольної фрази.

Асоційованим відкритим ключем можна вільно користуватися без будь-яких негативних наслідків. Відкритий ключ може використовуватися для шифрування повідомлень, які може розшифрувати тільки особистий ключ. Ця властивість використовується як спосіб автентифікації за допомогою пари ключів [54]. Схема роботи ключів наведена на рис. 3.3.

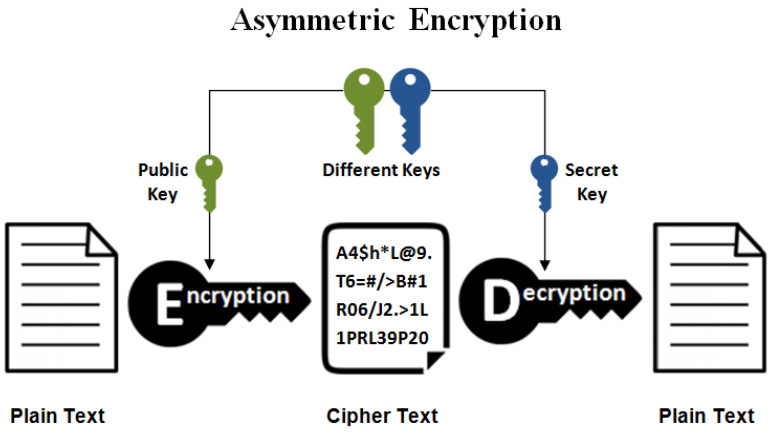


Рисунок 3.3 – Схема роботи приватного та відкритого ключів

Після створення таких ключів, можна завантажити їх в особистий профіль на GitHub, після чого, при взаємодії з проектом за допомогою git-команд, можна пройти процес аутентифікації для подальшої роботи з проектом.

Для роботи серверної частини, було обрано платформу NodeJs v8.11.4, що являє собою платформу для роботи з серверною частиною проекту, на основі мови програмування JavaScript.

Для завантаження пакетів розробки, було обрано пакетний менеджер NPM, що є інтегрованим в NodeJs, при встановлені. Окрім можливості завантаження пакетів-модулів, для виконання різноманітних задач, NPM має можливість зберігати інформацію про використані пакети в окремому файлі, контролювати їх версію, для підтримки сумісності з іншими технологіями проекту, а також створювати власні пакети чи команди, для оптимізації робочого процесу та подальшого використання їх в проекті чи експорті для застосування в інших проектах. Схема роботи пакетного менеджеру наведена на рис. 3.4.

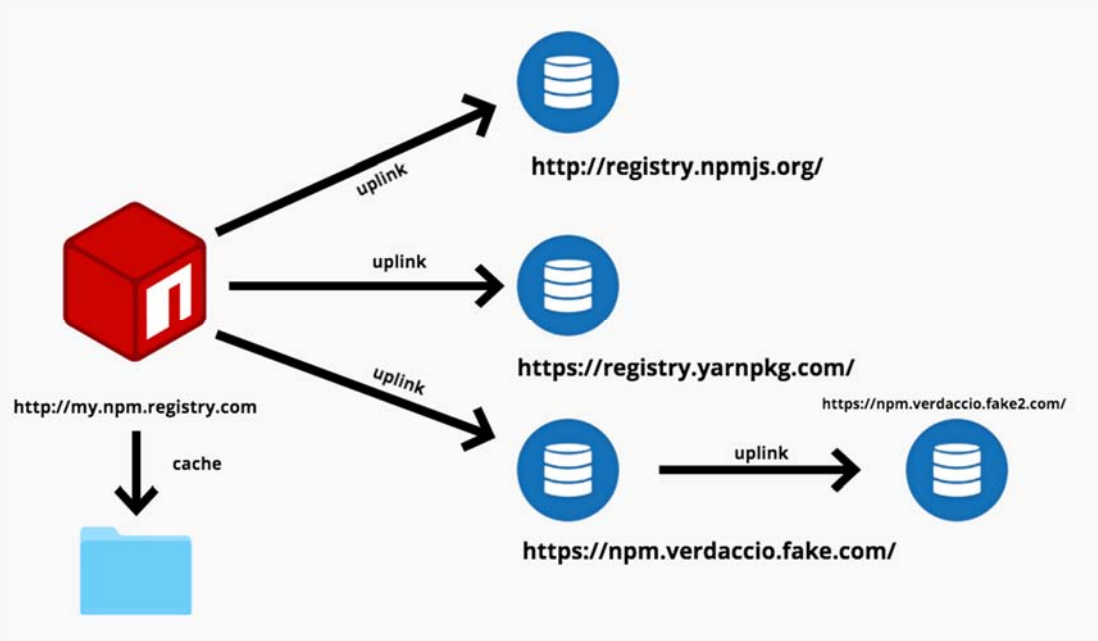


Рисунок 3.4 – Схема роботи пакетного менеджеру NPM

Для роботи з шаблонами сторінок, з використанням HTML, було обрано мову EJS. Це проста мова шаблонів, яка дозволяє генерувати розмітку HTML за допомогою звичайного JavaScript [55].

В якості веб-серверу, на якому буде працювати програмний додаток, було обрано сервер Express.

3.2 Опис технологічної платформи

Коли користувач формує запитує до клієнтської частини програмного додатку, відбувається наступне: формується структурований запит, який спочатку відправляється на сервер. Сервер оброблює або обчислює, робить перевірки запиту на стороні клієнта, і після виконання всіх таких перевірок відповідь відправляється знову на клієнтську частину. Для виконання подібних обчислень та обробки використовується фреймворк JavaScript NodeJS. У менеджері пакетів доступні понад 50 000 пакетів, тому розробники можуть імпортувати будь-який з них у будь-який час відповідно до необхідної їм функціональності, що значно економить час.

Оскільки NodeJS не потрібно чекати, поки API поверне дані, тому для побудови веб-додатків, що працюють в режимі реального часу та інтенсивно використовують дані, він є дуже корисним. Він повністю асинхронний за своєю природою, що означає, що він повністю не блокується.

Час завантаження аудіо або відео скорочується завдяки NodeJS, оскільки відбувається краща синхронізація коду між клієнтом і сервером через наявність однакової кодової бази.

Оскільки NodeJS має відкритий вихідний код і є нічим іншим, як фреймворком JavaScript, то для розробників, які вже звикли до JavaScript, почати розробляти свої проекти на NodeJs дуже легко [56].

Платформа NodeJS являє собою асинхронне кероване подіями середовище виконання JavaScript, призначене для створення масштабованих мережевих додатків. Під асинхронними тут маються на увазі всі ті функції в JavaScript, які обробляються у фоновому режимі, не блокуючи ніяких інших запитів.

NodeJS обробляє запити за допомогою циклу обробки подій всередині середовища NodeJS.

Цикл обробки подій називається слухач подій, який функціонує всередині середовища NodeJS і завжди готовий до прослуховування, обробки та видачі події [57]. Внутрішня побудова середовища наведена на рис. 3.5.

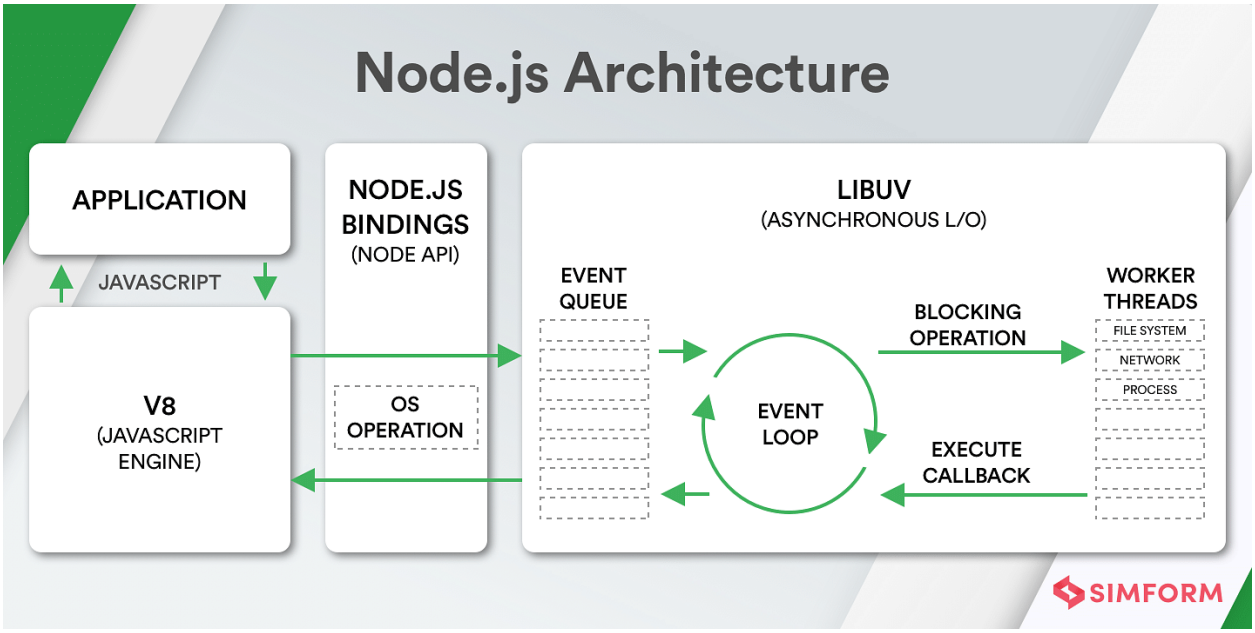


Рисунок 3.5 – Внутрішня архітектура NodeJs

Обробка даних на сервер – це основна частина програми. Він потребує великої уваги при його кодуванні. Якщо сервером не керувати або зробити неправильне його налаштування, то додаток буде наповнений ускладненнями та помилками, що зменшить потенційну швидкодію та ефективність додатку.

В якості серверної технології, було обрано технологію Express.Js. Основний напрямок його використання – це створення Restful API, які приймають запит від клієнта на сервер і відправляють відповідну відповідь [58]. Express.js – це бібліотека JavaScript, за допомогою якої ми можемо зручно працювати з налаштуванням серверної частини, не прибігаючи до використання громіздких технологій, типу Apache2, Nginx.

Вибір також зумовлений тим, що необхідно використовувати мову як для розробки клієнтської частини, так і серверу – JavaScript. Окрім цього, Express.js має офіційну підтримку технології Node.js, що покращує сумісність технологій та виключає проблеми при їх налагоджені. Завантажити серверну технологію можливо за допомогою пакетного менеджеру NPM.

3.3 Опис використаних бібліотек

Пакетний менеджер NPM дозволяє легко встановлювати необхідні пакети або модулі для роботи з різними сутностями, розширення функціоналу

проекту або для спрощення розробки, без необхідності створювати функціонал з нуля, не витрачаючи час та людські ресурси під час розробки додатку.

Для встановлення пакетів та налаштування команд в проекті, необхідно створити конфігураційний файл, в якому зберігається інформація про проект, посилання на Git-репозиторій, список команд, які можна використовувати для роботи з проектом, інформація про версію проекту і т.д. Одна з головних опцій конфігураційного файлу – інформація про використані бібліотеки та залежності, які необхідні для їх роботи. Розробник може вказати їх напряму в файлі або за допомогою синтаксису технології NPM в терміналі системи.

Окрім бібліотек, які використовуються в проекті на етапі розробки та ввід в експлуатацію, існують також залежності, які необхідні під час розробки додатку. Вони спрощують розробку, дозволяють проводити діагностування помилок, виведення більш детальної інформації про ресурси, дозволяють проводити тестування окремих елементів системи і т.д. В якості допоміжних бібліотек було обрано такі бібліотеки як Gulp, Nodemon, Sass які будуть встановлені в проекті за допомогою пакетного менеджеру NPM.

Gulp або Gulp.js – це потужний інструмент для запуску команд та задач, що використовує Node.js в якості платформи. Gulp працює виключено з JavaScript та допомагає запускати задачі для клієнтської сторони веб-додатків. Він створює системні автоматизовані завдання, такі як спрощення та мінімізація коду в файлах CSS та HTML, конкатенація бібліотечних файлів та компіляція SASS-файлів. Завдання можна запускати через термінал [59].

Nodemon – інструмент для розробки додатків на основі Node.js з метою автоматизації та спрощення розробки додатку. Інструмент дозволяє запустити фоновий процес, який відслідковує зміни в поточному проекті та перезавантажує його для прийняття змін, економлячи час розробника, дозволяючи зосередитись на вирішенні задач. [60].

Sass – інструмент для перетворення файлів .sass або .scss на файли CSS. Окрім форматування файлів, інструмент дозволяє запустити процес, щоб

відслідковувати зміни в робочих файлах, на аналог з Nodemon. Також інструмент дозволяє використовувати налаштування при форматуванні файлів, такі як спрощення синтаксису та оптимізація поточних файлів, а також шлях до директорії, куди будуть зберігатись форматовані файли, які вже будуть використовуватись в самому проекті. Інструмент дозволяє використовувати змінні, вкладені правила, міксини, функції, за допомогою повністю сумісного з CSS синтаксису [61].

Всі ці інструменти необхідні для розробки та при введені в експлуатацію, для ініціалізації готового проекту. Кожен інструмент використовується пакетним менеджером, для швидкого доступу та виконання тих задач, для яких інструмент призначений.

Для функціонування проекту, необхідно використання основних модулів, опис яких буде наведений нижче.

Bootstrap – популярний інструментарій, потужний, масштабований, а також багатофункціональний, для роботи з клієнтською частиною [62]. Інструментарій дозволяє використовувати власну сітку для розташування HTML-елементів для адаптації під різні пристрої, має власну систему стилів, яку можна швидко інтегрувати в проект та використовувати для різних елементів, а також власні бібліотеки для створення інтерактивних елементів сторінки. Сітка Bootstrap використовує ряд контейнерів, рядків і стовпців для організації та вирівнювання вмісту. Вона побудована за допомогою flexbox-елементів і являється повністю адаптивною [63].

Chart.js – JavaScript бібліотека для створення графіків на основі HTML. Бібліотека дозволяє створювати різні графіки та діаграми, для візуалізації даних: гістограми, кругова, лінійна, розсіювання, зональна, радарна діаграма і т.д [64]. Окрім самої бібліотеки, також встановлені додаткові пакети для більш гнучкого налаштування діаграми.

Ejs – движок шаблонів, або шаблонізатор, який може використовуватись в Node.js. Основне призначення – можливість простої інтеграції даних в

HTML-шаблон на клієнтській частині для форматування в кінцевий HTML [65].

Express або Express.js – серверна основа програми, детальну інформацію про яку було наведено вище, в описі технологічної платформи. Окрім серверної основи, також встановлена допоміжна бібліотека до нього, для завантаження файлів, які необхідні для дослідження та аналізу даних.

jQuery – швидка, компактна, багатофункціональна бібліотека JavaScript, яка значно спрощує роботу з DOM-деревом, дозволяє такі речі, як маніпуляції з HTML-документами, обробка подій, анімація, використання технології Ajax, забезпечує підтримку в багатьох браузерах [66].

Jsdom – бібліотека для роботи з DOM-елементами сторінки. Реалізація виконана на JavaScript багатьох веб-стандартів, для використання з Node.js. Основна мета проекту – емуляція достатньої підмножини веб-браузерів, для для тестування, парсингу реальних веб-додатків [67].

Описані бібліотеки та їх актуальні версії на проекті записані та збережені в конфігураційному файлі пакетного менеджеру NPM. У разі необхідності використання іншої версії або оновлення бібліотеки, можна змінити версію або в самому конфігураційному файлі або за допомогою команд в терміналі. Актуальні робочі версії бібліотек проекту і сам файл конфігурації також занесені в Git-репозиторій.

3.4 Проектування та розробка програмного забезпечення

Додаток має бути розділений на частини, які будуть взаємодіяти між собою. Кожен з них має виконувати одну конкретну дію – по принципу єдиної відповідальності SOLID, коли кожен клас має виконувати лише одну і тільки одну дію [68]. Можна виділити три частини програми, які будуть відокремлені один від одного.

Перша частина відповідає за взаємодію з користувачем – сторінка з формою для завантаження документу, обробка та відправка на сервер, отримання результатів та їх перегляд.

Друга частина відповідає за роботу сервера. Він має запускати необхідні залежності для роботи бібліотек, налаштовувати стилі та скрипти, отримувати, обробляти та відправляти відповідь на запити.

Третя частина відповідає за роботу модулів. Модулі працюють за єдиним принципом – отримують сторінку, виконують внутрішню логіку, формують результат аналізу та відправляють на сервер, за його запитом.

Для візуалізації процесу роботи частин між собою, необхідно побудувати діаграму активності, див рисунок 3.6. Діаграма активності використовується для ілюстрації потоку управління в системі та посилання на кроки, пов'язані з виконанням варіанту використання, а також для моделювання послідовних та паралельних дії [69]. Діаграма активності частин додатку наведена на рис. 3.6.

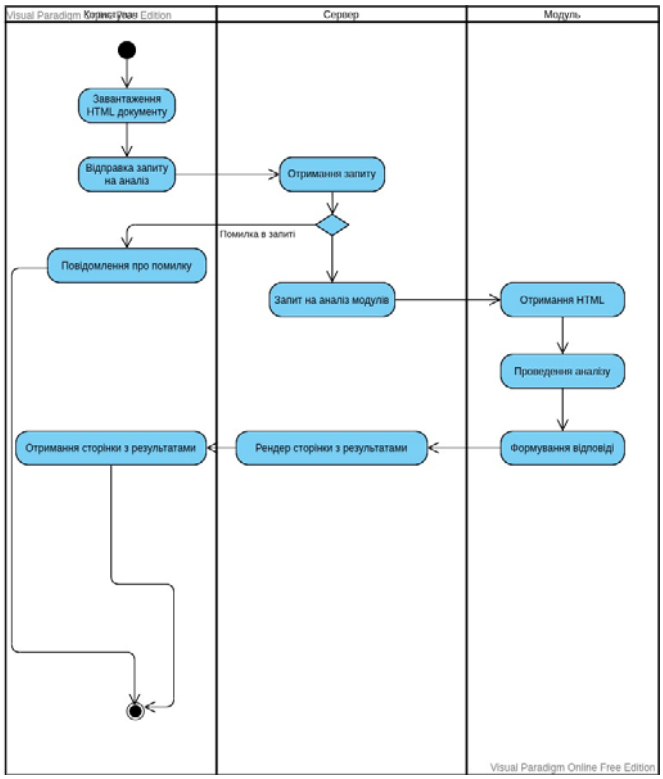


Рисунок 3.6 – Діаграма активності взаємодії складових частин додатку

Для початку роботи, необхідно ініціалізувати проект, для подальшого його збереження в Git-репозиторій. Для цього необхідно створити аккаунт на платформі GitHub та створити репозиторій – на етапі розробки, репозиторій буде приватним, тобто, не буде доступний для зовнішнього перегляду.

Після створення репозиторію, необхідно вказати точку в системі, де буде знаходитись проект. Після ініціалізації проекту, в директорії проекту буде створений файл .git, який відслідковує етапи розробки та дозволяє продивлятись статус репозиторію. Після створення точки проекту, буде створена основна вітка проекту – master. В ній будуть зберігатись актуальна версія проекту. Термінал перегляду статусу проекту наведений на рис. 3.7.

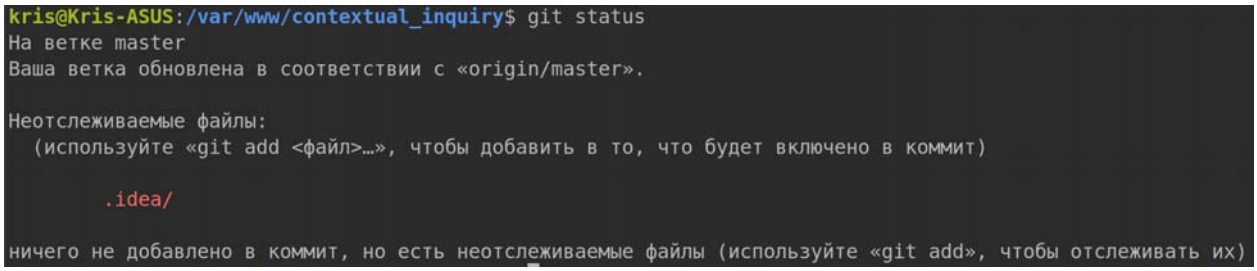


Рисунок 3.7 – Термінал перегляду статусу вітки master

Після цього, необхідно створити конфігураційний файл пакетного менеджеру package.json та вказати всі необхідні залежності, необхідні як для функціонування проекту, так і для розробки. В файлі вкажемо інформацію про проект, автора проекту, заснемо всі залежності, необхідні для подальшої розробки. Після налаштування конфігураційного файлу, запустимо процес встановлення пакетів, за допомогою інструментів пакетного менеджеру. Файл конфігурації зберігається в форматі JSON – всередину можна такі ж типи даних, що і в стандартний об'єкт JavaScript – рядки, числа, масиви, логічні значення та інші об'єктні літерали [70].

Точкою входу в проект буде файл index.js.

Зміст конфігураційного файлу наведений на рис. 3.8.

Рисунок 3.8 – Конфігураційний файл пакетного менеджеру

Для пакетного менеджеру, створимо додаткові команди, які будемо використовувати на моменті розробки додатку. Команди записуються в полі scripts, яке відповідає за виконання команд ОС, з можливістю їх автоматизації через інструментарій пакетного менеджеру.

Команда scss запускає процес оптимізації файлів стилів, форматуючи їх в компактний варіант, із збереженням в директорію, яку ми вказали в конфігураційному файлі. Флаг --watch встановлює, що команда буде запущена в фоні та буде запускати процес оптимізації та форматування файлів кожен раз, коли файли стилів будуть змінені.

Команда nodemon-dev запускає сервер Express, встановлюючи локальний доступ за адресою localhost:3000. Команда запущена в фоні та перезавантажується кожен раз, коли в файлах проекту будуть робитись зміни.

Команда dev – технічна, необхідна для закриття всіх процесів NodeJs та запуску серверу, у виникненні помилок при завантажені серверу.

На етапі налаштування проекту та встановлення всіх бібліотек, маємо шаблон для роботи з платформою NodeJs. Необхідно створити директорії файлів, з якими в подальшому будемо працювати з бібліотеками, стилями, скриптами і т.д.

Схема проекту наведена на рис. 3.9.

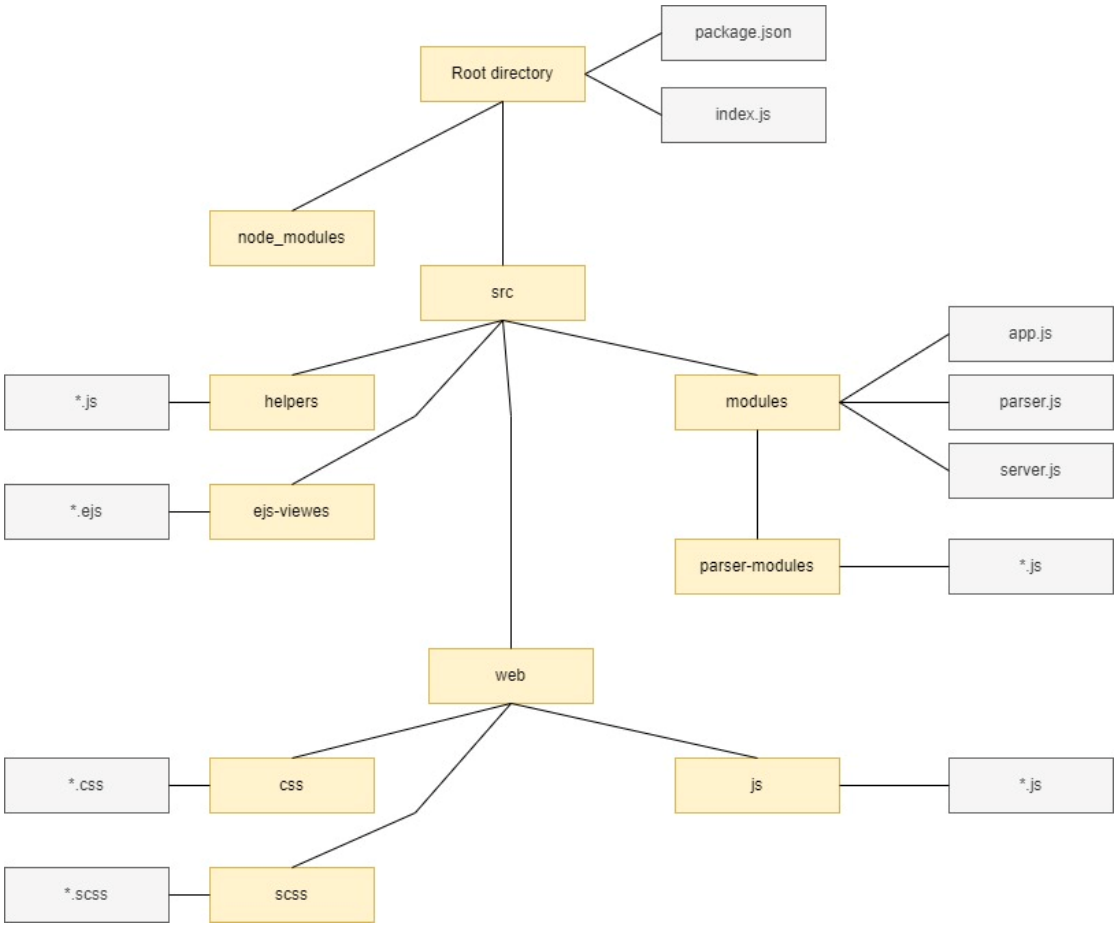


Рисунок 3.9 – Дерево файлової структури проекту

В директорії node_modules зберігаються всі файли залежностей та бібліотек, які ми вказували в конфігураційному файлі пакетного менеджеру. Вона створюється після інсталяції залежностей і не індексується Git-репозиторієм, через великий розмір файлів.

В директорії src знаходяться всі файли та модулі програмного додатку. Всі вони розділені між собою на складові частини, для кращої навігації в проекті.

В директорії ejs-views знаходяться всі сторінки HTML, з динамічними елементами Javascript, в форматі .ejs. Приклад представлення наведений на рис. 3.10.

```
<head>

    <script src="js/jquery.min.js"></script>
    <script src="js/index.js"></script>

    <link rel="stylesheet" href="css/bootstrap.min.css">
    <link rel="stylesheet" href="css/index.css">

    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Web-App | Main</title>
</head>
<body>
<div class="body-index">
    <div class="content-wrapper">
        <div class="text-wrapper">
            <h2 class="app-text">Перший веб-додаток з використанням методу контекстного до
            <h2 class="process-text">Запущен процесс...</h2>
        </div>
        <div class="form-wrapper">
            <form id="upload-form" action="/upload" method="post" enctype="multipart/form-
                <label for="apply"><input type="file" name="document" id="apply" accept=".
                <p class="form-text">Тільки *.html підтримуються</p>
            </form>
        </div>
    </div>
</div>
</body>
```

Рисунок 3.10 – Внутрішнє представлення файлу .ejs

В директорії helpers знаходяться допоміжні файли, які використовуються в різних частинах проекту. В них зберігається код, який є часто використовуваним і слугує як технічна бібліотека, для виконання певного функціоналу. Файли зберігається в форматі .js.

В директорії modules зберігаються модулі, за якими проходить аналіз сторінок. Всі описані модулі, що виконують аналіз за визначеними критеріями, зберігаються в директорії parser-modules, в форматі .js.

Файл app.js зберігається клас додатку, який запускає шаблонізатор сторінок, запускає обробку запитів, ініціалізує файли стилів та скрипти, а також запускає сервер.

Файл server.js являє собою клас серверу, який налаштовує необхідні стилі та скрипти, для роботи на стороні клієнту, встановлює правила та логіку для обробки запитів, налаштовує шаблонізатор.

Файл parser.js являє собою клас парсеру, який запускає в роботу аналізатор модулів. Зберігає в собі список тих модулів, які необхідно запустити та видає кінцевий результат обробки, який потім передається на сторінку користувачу. Діаграма класів наведена на рис. 3.11.

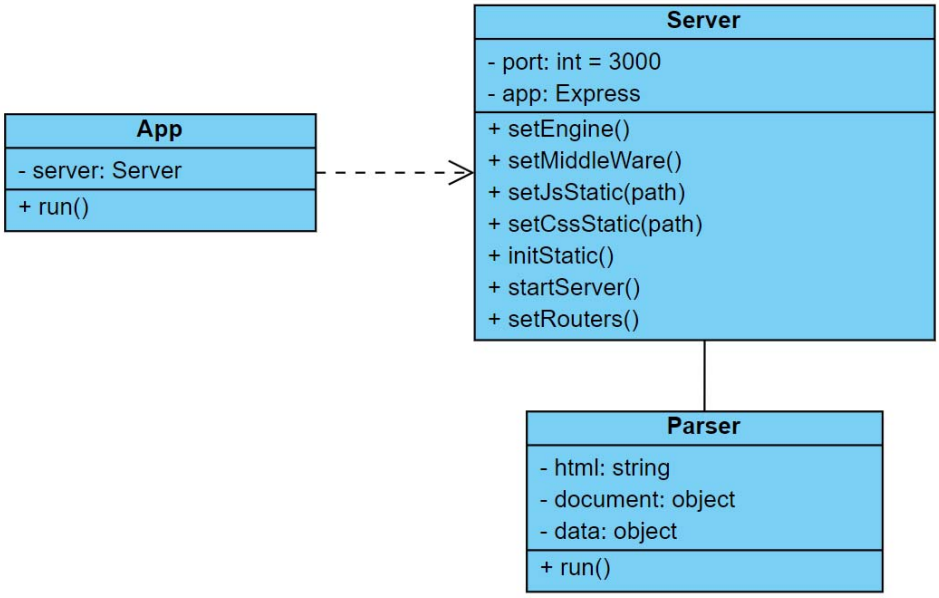


Рисунок 3.11 – Діаграма класів веб-додатку

В директорії web зберігаються файли, які завантажуються на сторінці користувача. Так, в директорії scss зберігаються файли стилів, в форматі .scss, які після форматування пакетним менеджером, форматуються в формат .css та зберігаються в директорії css відповідно, які вже завантажуються на сторінці. В директорії js зберігаються основні скрипти в форматі .js, які завантажуються на сторінці користувача і дозволяють створювати інтерактивні елементи.

3.5 Опис елементів системи

В момент запуску додатку, завантажується головна сторінка програми. На ній представлено текст з описом програмного додатку, кнопка для завантаження файлу формату .html. Сторінка супроводжується текстом з поясненням функціонального призначення кнопки та з нагадуванням, що

наразі підтримується тільки формат .html. Головну сторінку наведено на рис. 3.12.

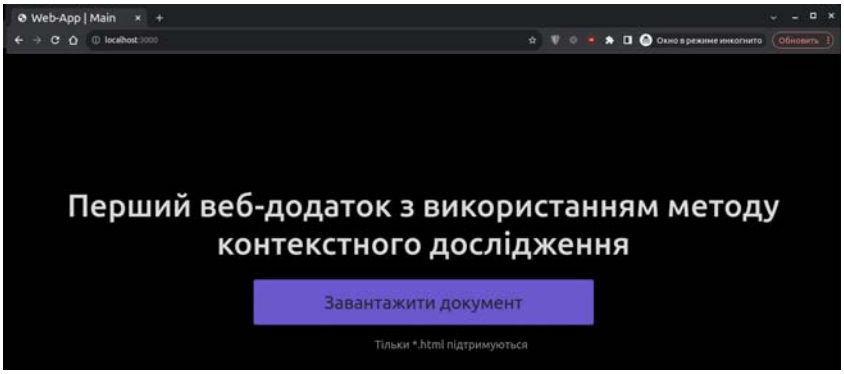


Рисунок 3.12 – Головна сторінка програми

Після натискання на кнопку «Завантажити документ», відкривається модальне вікно з провідником системи, для вибору файлу для завантаження. В залежності від системи, вікно інтерфейсу провідника може відрізнятись. В провіднику встановлений фільтр, який обмежує формат файлу – тільки .html. Екранну форму провідника системи наведено на рис. 3.13.

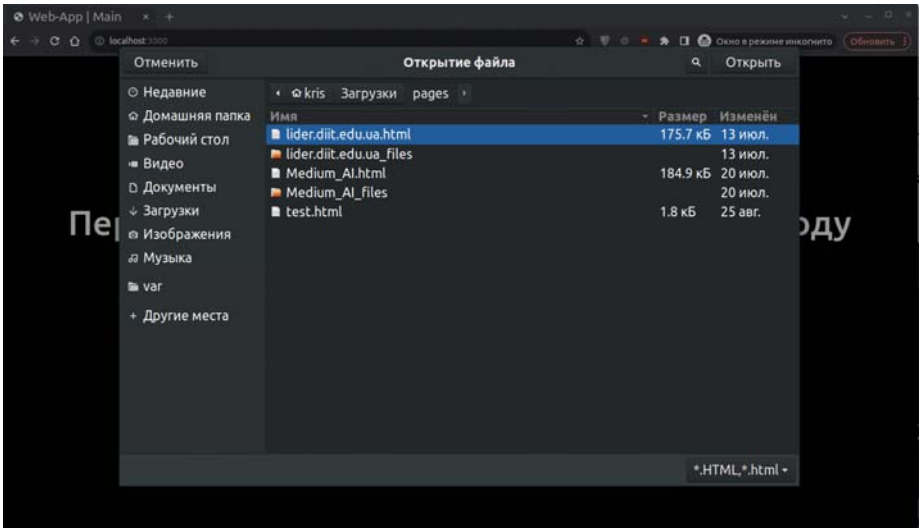


Рисунок 3.13 – Екранна форма провідника системи

Необхідно обрати файл, який буде досліджено, необхідного формату. У випадку, якщо формат обрано неправильно або є спроба завантаження непідтримуваного файлу – система видає відповідну помилку, інформуючи користувача, що він зробив неправильний крок. Приклад повідомлення про помилку наведений на рис. 3.14.

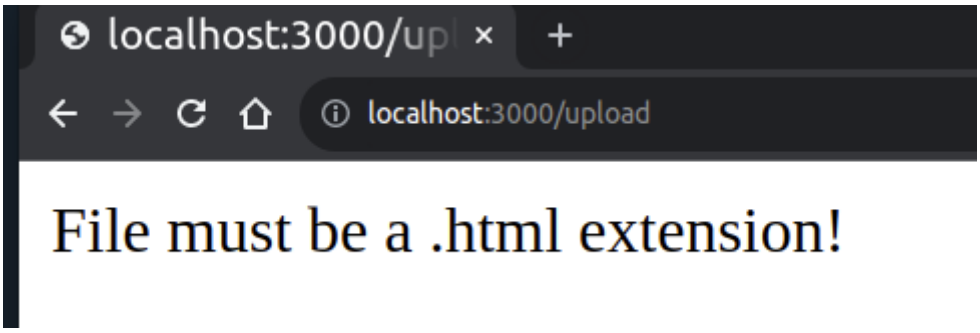


Рисунок 3.14 – Повідомлення про помилку при завантажені неправильного формату файлу

Після вибору файлу підтримуваного формату та підтвердження вибору, файл відправляється на сервер, для аналізу. В момент, поки йде аналіз, користувачу виводиться відповідне повідомлення. Кнопка завантаження документу перестає бути активною, для запобігання помилок. Сторінка після завантаження файлу наведена на рис. 3.15.

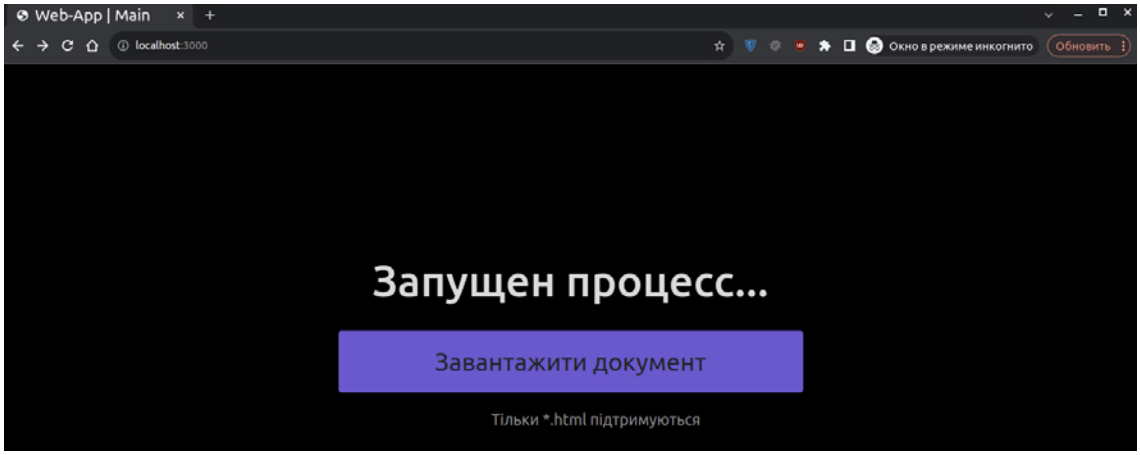


Рисунок 3.15 – Головна сторінка після завантаження файлу

В залежності від складності та розміру файлу, аналіз сторінки займає різний проміжок часу. Коли аналіз завершиться, сторінка перенаправляє користувача на сторінку з результатами аналізу.

На сторінці з аналізом представлені діаграми з модулями контекстного аналізу, які наразі прописані в проекті. Дані та наповнення сторінки можуть відрізнятись, в залежності від файлу, який був завантажений. Сторінка з результатом аналізу наведено на рис. 3.16.

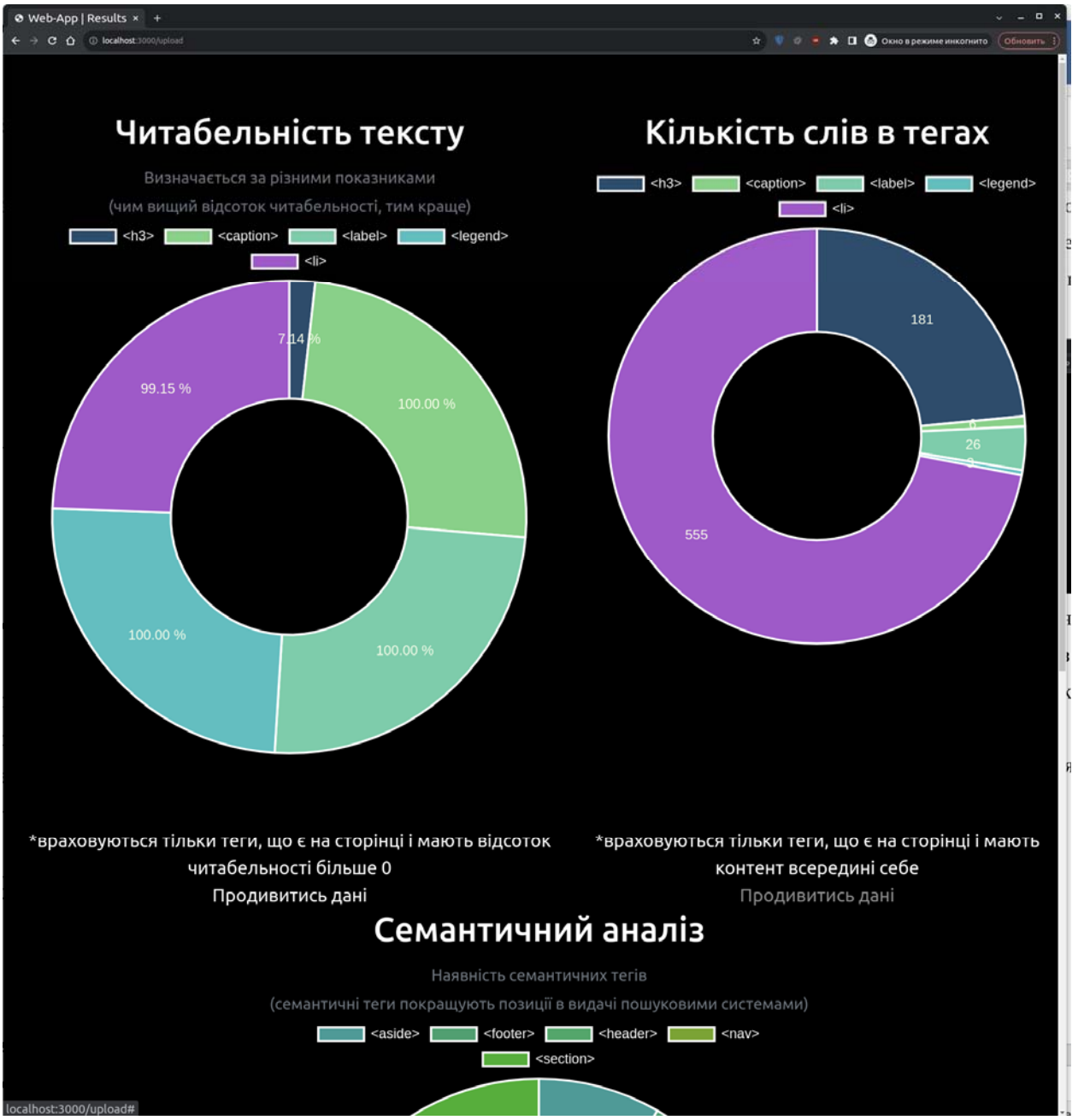


Рисунок 3.16 – Сторінка з результатами аналізу файлу

Основним елементом на сторінці виступає кругова діаграма, яка дозволяє отримані дані візуалізувати, для кращого сприйняття. Кожна діаграма відповідає за окремий модуль, через який проходить аналіз переданого на сервер файлу.

Кожен модуль складається з таких частин:

1) Назва модуля.

2) Опис модуля. Пояснює функціональне призначення модуля. Може бути відсутній.

3) Додатковий опис. Додаткова інформація до опису модуля. Може бути відсутній.

4) Легенда діаграми. Перелік елементів (в даному випадку – тегів), які відображені на круговій діаграмі. Являються інтерактивними – при натисканні на елемент легенди, дозволяють відобразити або приховати елемент з діаграми, що супроводжується анімацією. Окрім цього, деякі діаграми зв’язані між собою, якщо це передбачено в модулі – інтерактивні елементи, при взаємодії з ними, будуть застосовані до всіх зв’язаних діаграм.

5) Діаграма.

6) Примітка до даних. Може бути відсутня.

7) Кнопка для перегляду даних. Відкриває модальне вікно, з можливістю перегляду даних, які були оброблені модулем.

Приклад діаграми на сторінці результатів наведено на рис. 3.17.

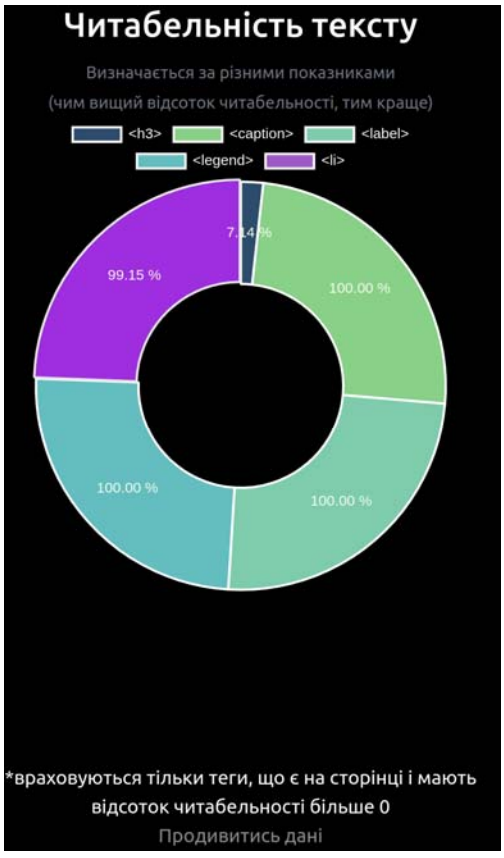


Рисунок 3.17 – Діаграма аналізу модуля читабельності тексту

В разі необхідності переглянути дані модулю, необхідно натиснути на надпис «Продивитись дані» – відкриється модальне вікно, в якому буде представлена детальна інформація по даним, у вигляді дерева, з яких будувалась кругова діаграма. При натисканні на елементи дерева, будуть відкриватись або приховуватись елементи дерева. В даному випадку, при натисканні на надпис для модулю читабельності тексту – представлена інформація про коефіцієнт читабельності тексту для всіх тегів, визначених в модулі. В залежності від модулю, дані можуть відрізнятись. Екранна форма з даними модуля представлена на рис. 3.18.

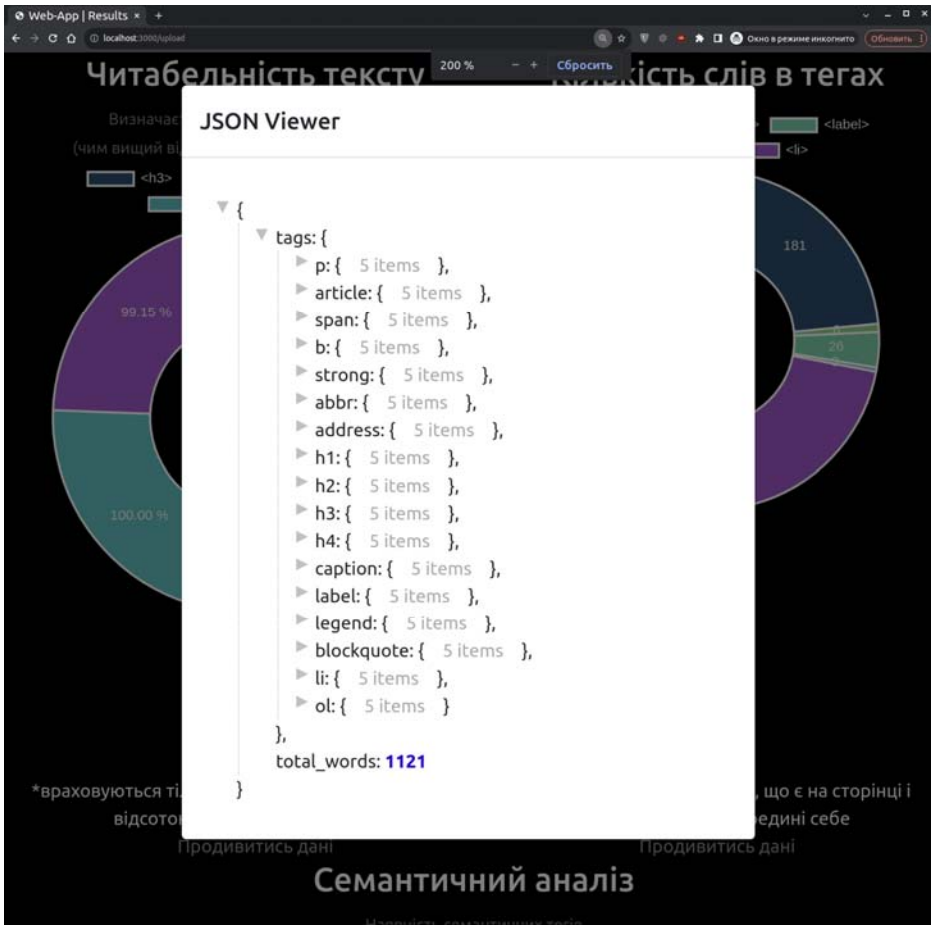


Рисунок 3.18 – Екранна форма перегляду даних модуля

На прикладі читабельності тексту, можна переглянути як загальну інформацію про читабельність за тегом, так і за окремим елементом, в якому зберігається вся інформація, необхідна для аналізу користувачу, навіть якщо

елемент не відображено на графіку. Формат даних модуля представлений на рис. 3.19.

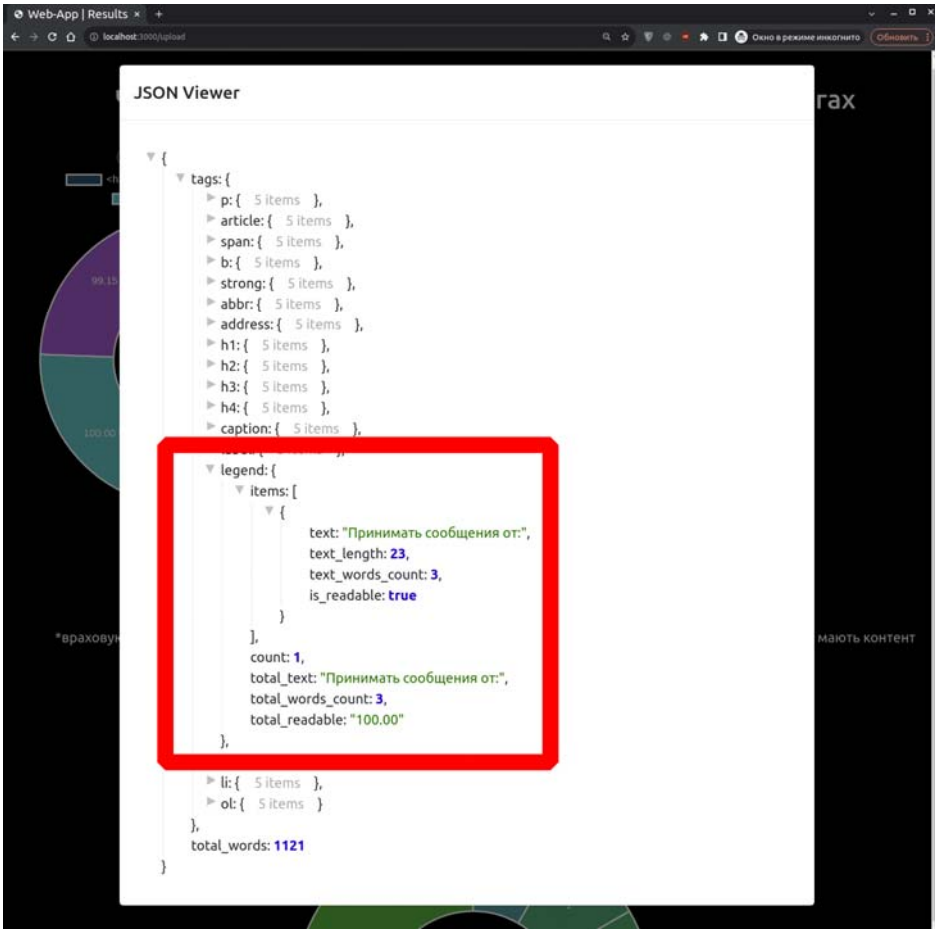


Рисунок 3.19 – Екранна форма перегляду даних по тегу

3.6 Висновки

Виконано проектування архітектури інструментального забезпечення для дослідження веб-сайту за допомогою методу контекстних досліджень. У ході проектування, було передбачено можливість модернізації та розширення системи за рахунок відокремлення модулів для аналізу сторінок в окремій директорії та пов’язаних спільною логікою класами, а також винесення проекту в окремий Git-репозиторій для можливості подальшого вдосконалення та модернізації. Відповідно до принципу єдиної відповідальності, кожен клас системи має тільки один обов’язок. Це спрощує сприйняття коду програми, покращує навігацію по проекту, дозволяє легше орієнтуватись по його структурним елементам.

У розділі наведено опис середовища для розробки інструментального забезпечення, описані підходи до проектування та вибір мови програмування, пов'язаних технологій. Наведений детальний опис використовуваних технологій в проекті: аргументовано вибір таких технологій, описано принцип їх роботи, особливості та функціональне призначення в проекті.

У розділі наведено опис проектування інтерфейсу користувача. Наведені основні елементи системи, їх структурне та функціональне призначення.

Завдяки використанню технологій веб-додатків, систему можна використовувати під керуванням різних операційних систем (Microsoft Windows, Ubuntu Linux, Mac OS), де присутній веб-браузер.

Система готова до впровадження та використання.

4 ДОСЛІДЖЕННЯ ВЕБ-САЙТІВ

4.1 Підготовка до експерименту

Перед початком експерименту, необхідно перевірити стабільність системи, на якій проводиться тестування, актуальність роботи бібліотек та залежностей в пакетному менеджері.

Експеримент повинен показати – чи можна використовувати метод контекстних досліджень за допомогою розробленого інструментального засобу, на основі результатів – зробити висновки щодо доцільності такого використання, з результативних даних зробити висновок щодо ефективності використання методу контекстних досліджень та розробленого інструментарію.

4.2 Проведення експерименту

4.2.1 Експеримент 1

Мета: Провести експеримент по аналізу сторінки всесвітньої газети The Times.

Вхідні дані – html файл головної сторінки веб-сайту, зображення якої наведено на рис. 4.1 та внутрішнє представлення на рис. 4.2.

Рисунок 4.1 – Головна сторінка The Times

Рисунок 4.2 – Внутрішнє представлення сторінки The Times

Результати експерименту наведені на рис. 4.3 - 4.4.

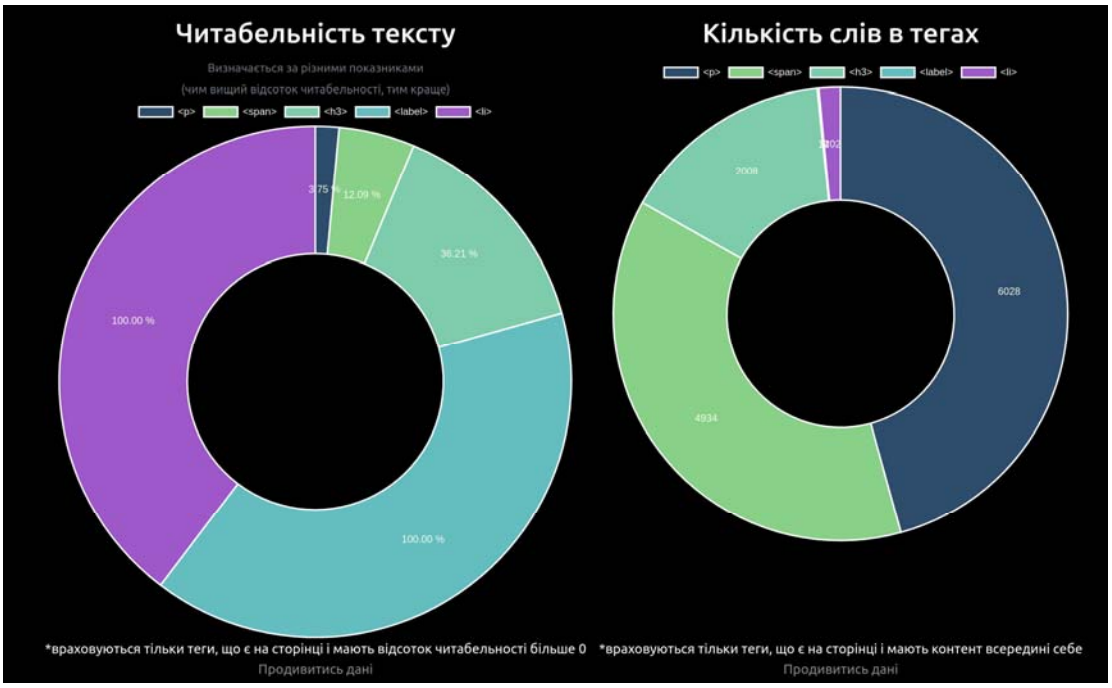


Рисунок 4.3 – Текстовий аналіз сторінки The Times

На графіках можна побачити, що більша частина тексту припадає на параграфи, строки та заголовки третього рівня. При цьому, найбільший відсоток читабельності припадає підписи до елементів та спискові елементи, найменші – на параграфи та строки.

Таблиця 4.1 – Текстовий аналіз сторінки The Times

Назва елементу	Кількість слів	Читабельність, %
<p>	6028	3.75
	4934	12.09
<h3>	2008	36.21
<label>	14	100
	202	100

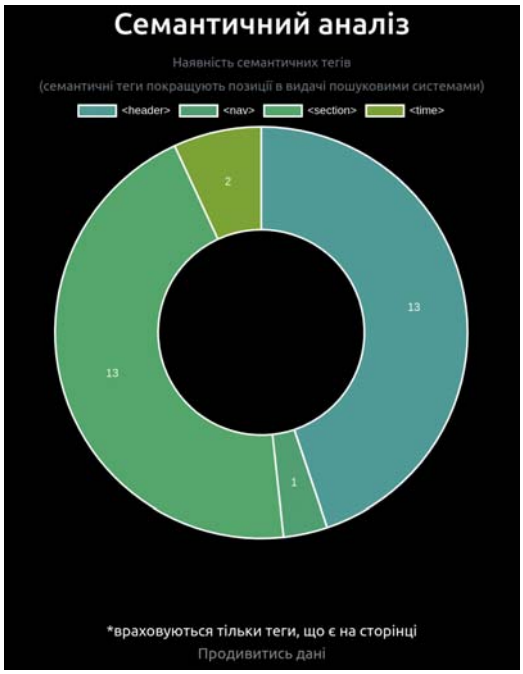


Рисунок 4.4 – Семантичний аналіз сторінки The Times

Наявні основні теги для сторінки, є розділення на секції, для навігації по окремим елементам сторінки.

Таблиця 4.2 – Семантичний аналіз сторінки The Times

Назва елементу	Кількість
<header>	13
<nav>	1
<section>	13
<time>	2

Висновки: на основі текстового аналізу, можна виділити, що теги з низькою читабельністю використовуються недоцільно до контексту. Так, параграфи використані для запису одного або пари слів, хоча доцільніше було б використовувати інші теги, для їх запису – strong, span, b чи інші. Детальне представлення даних наведено на рис. 4.5.

Рисунок 4.5 – Аналіз читабельності параграфів газети The Times

Семантичний аналіз демонструє достатній рівень для аналізу сторінки пошуковими системами, що гарантує високу вірогідність індексації сторінки при пошукових запитах.

4.2.2 Експеримент 2

Мета: Провести експеримент по аналізу головної сторінки української онлайн-газети ГОРДОН.

Вхідні дані – html файл головної сторінки веб-сайту, зображення якої наведено на рис. 4.6 та внутрішнє представлення на рис. 4.7.

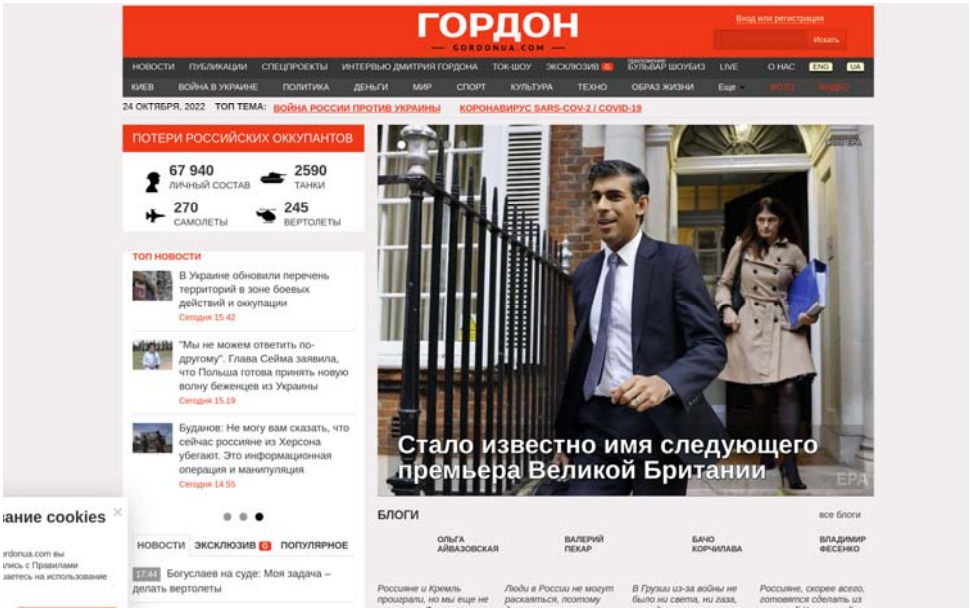


Рисунок 4.6 – Головна сторінка ГОРДОН

Рисунок 4.7 – Внутрішнє представлення сторінки ГОРДОН

Результати експерименту наведені на рис. 4.8 - 4.9.

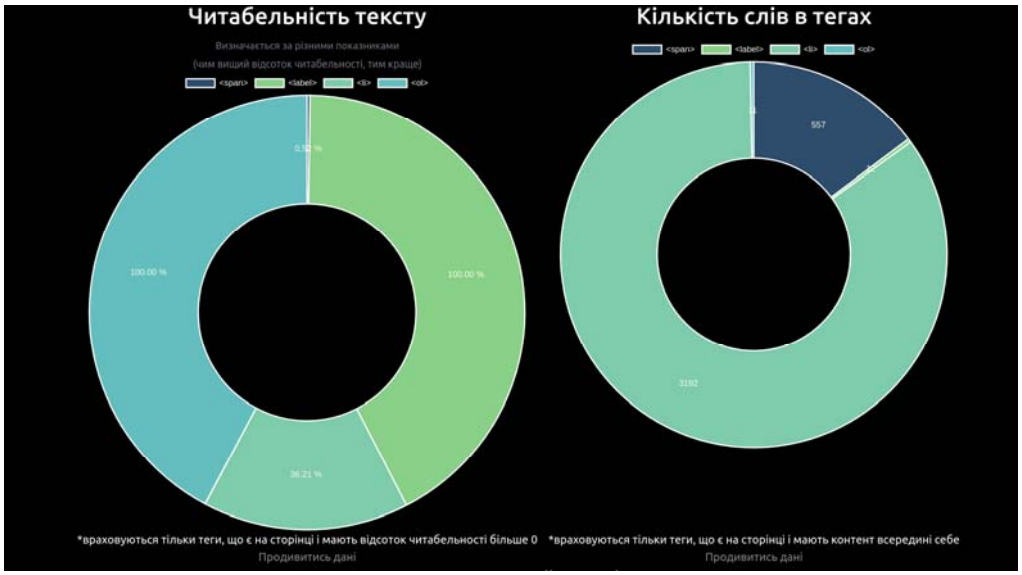


Рисунок 4.8 – Текстовий аналіз сторінки ГОРДОН

На графіках можна побачити, що більша частина тексту припадає на списки та строки. Підписи до елементів та упорядковані елементи списку складають найменший відсоток загальної кількості слів на сторінці, згідно графіку.

Таблиця 4.3 – Текстовий аналіз сторінки ГОРДОН

Назва елементу	Кількість слів	Читабельність, %
	557	0.52
<label>	12	100
	3192	36.21
	11	100

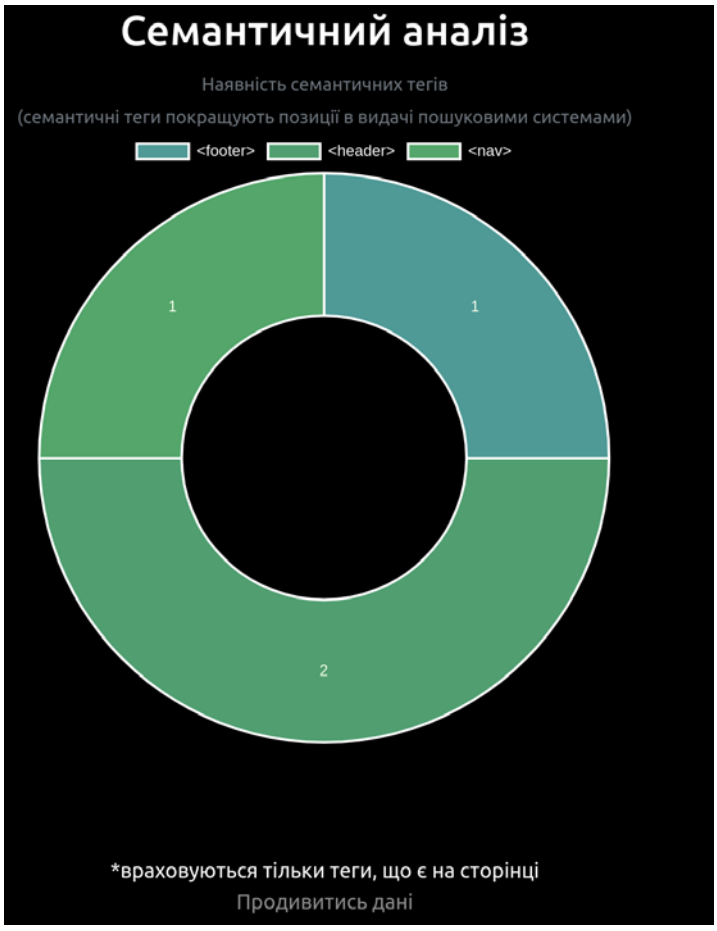


Рисунок 4.9 – Семантичний аналіз сторінки ГОРДОН

Наявний мінімальний набір тегів, для розділення сайту, для покращення навігації та індексації.

Таблиця 4.4 – Семантичний аналіз сторінки ГОРДОН

Назва елементу	Кількість
<footer>	1
<header>	2
<nav>	1

Висновки: текстовий аналіз демонструє максимальний рівень читабельності для підписів до елементів та упорядкованих елементів списку. Проте, строкові теги мають рівень читабельності близький до нуля – це пов’язано з відсутністю інших тегів на сторінці, таких як заголовки, параграфи, форматовані теги для тексту, які могли б використовуватись для покращення сприйняття сторінки. Це ж саме стосується і елементів списку – можна підвищити читабельність сторінки для елементів списків, розширивши кількість текстових тегів.

Семантичний аналіз демонструє мінімальний набір ключових тегів, що не сприяє великому відсотку індексації пошуковими системами, а також ускладнює навігацію по змісту сторінки, через відсутність розподілених секцій.

4.2.3 Експеримент 3

Мета: Провести експеримент по аналізу всесвітньої газети BBC News.

Вхідні дані – html файл головної сторінки веб-сайту, зображення якої наведено на рис. 4.10 та внутрішнє представлення на рис. 4.11.

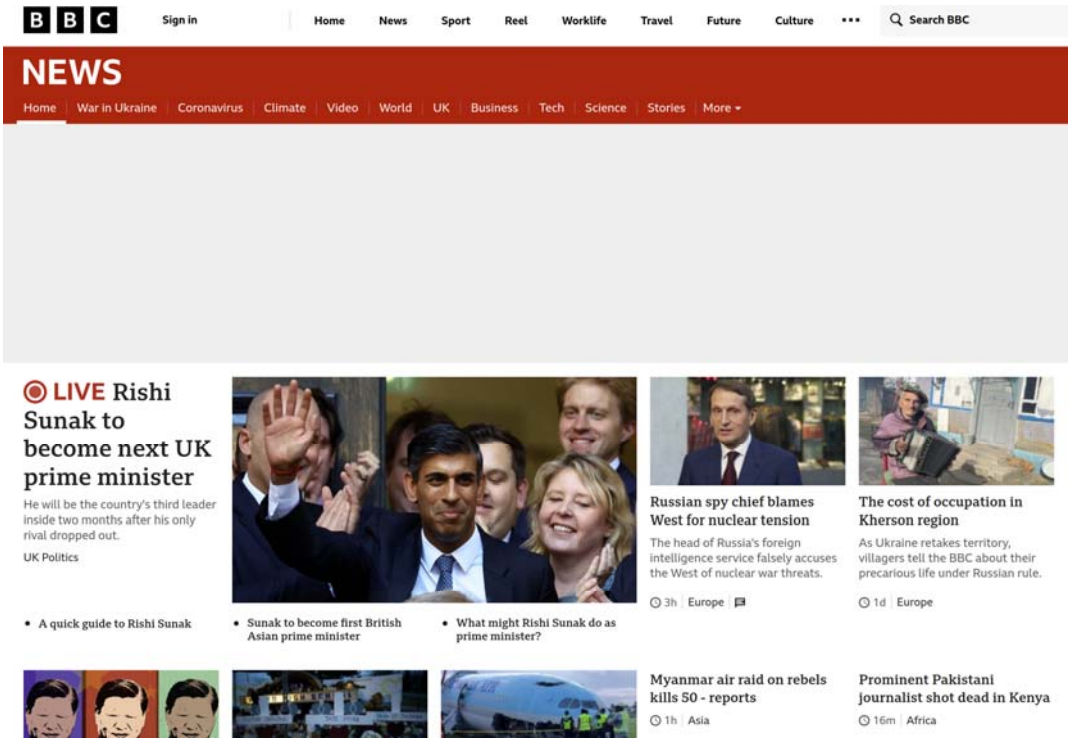


Рисунок 4.10 – Головна сторінка BBC News

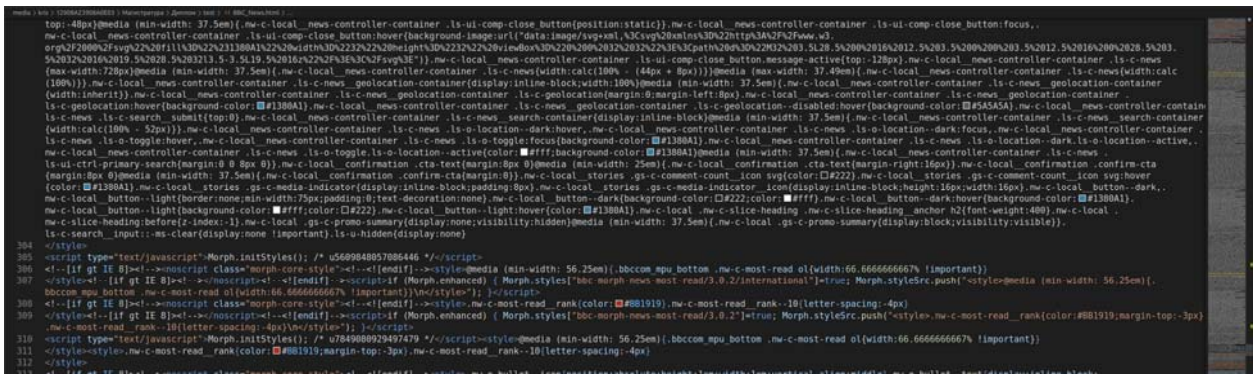


Рисунок 4.11 – Внутрішнє представлення сторінки BBC News

Результати експерименту наведені на рис. 4.12 - 4.13.

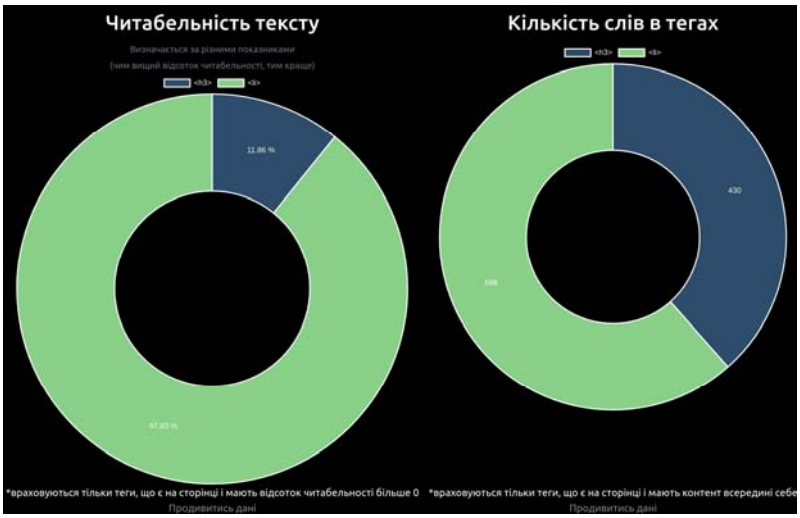


Рисунок 4.12 – Текстовий аналіз сторінки BBC News

За текстовим графіком, можемо побачити, що виділено лише два елементи – елементи списку та заголовки третього рівня. Інші текстові елементи відсутні.

Таблиця 4.5 – Текстовий аналіз сторінки BBC News

Назва елементу	Кількість слів	Читабельність, %
<h3>	430	11.86
	688	97.83

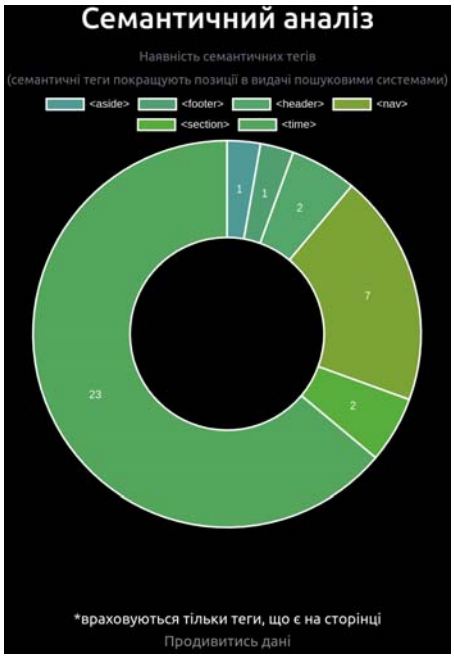


Рисунок 4.13 – Семантичний аналіз сторінки BBC News

На графіку спостерігаємо велику кількість різноманітних семантичних тегів, секції розділення вмісту сторінки.

Таблиця 4.6 – Семантичний аналіз сторінки BBC News

Назва елементу	Кількість
<aside>	1
<footer>	1
<header>	2
<nav>	7
<section>	2
<time>	23

Висновки: текстовий аналіз демонструє високу читабельність елементів списку, але невелике значення для заголовків. Для покращення читабельності, необхідно або замінити заголовки на інші текстові теги, або змінити їх структуру.

Семантичний аналіз демонструє високий рівень для аналізу сторінки пошуковими системами, що гарантує високу вірогідність індексації сторінки при пошукових запитах. Велика кількість семантичних тегів також покращує навігацію по вмісту веб-сайту для розробника.

4.2.4 Експеримент 4

Мета: Провести експеримент по аналізу інформаційної платформи IGN.

Вхідні дані – html файл головної сторінки веб-сайту, зображення якої наведено на рис. 4.14 та внутрішнє представлення на рис. 4.15.

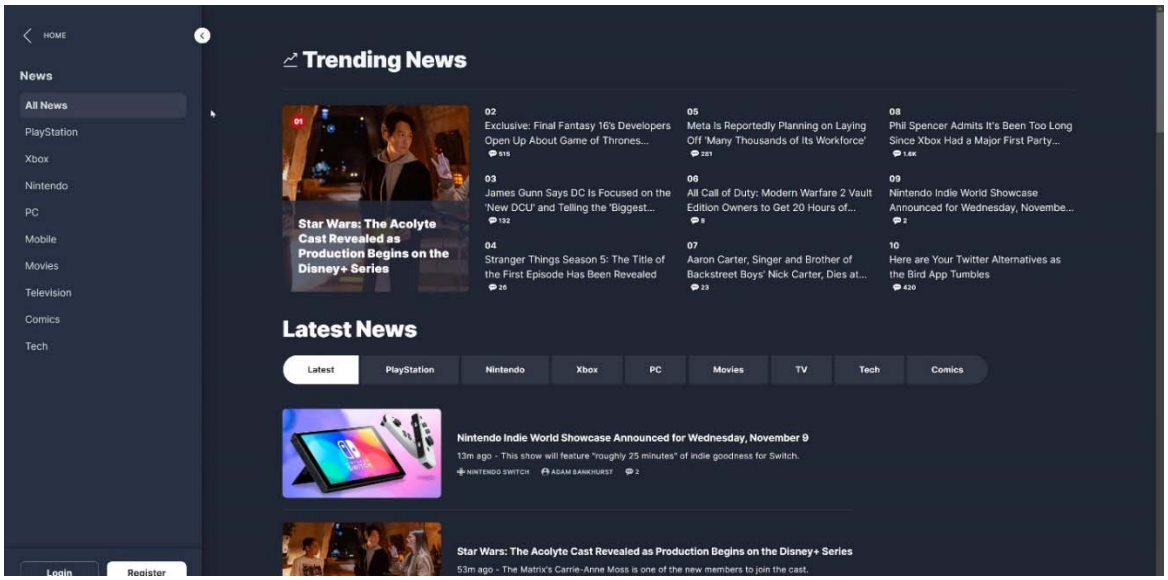


Рисунок 4.14 – Головна сторінка IGN

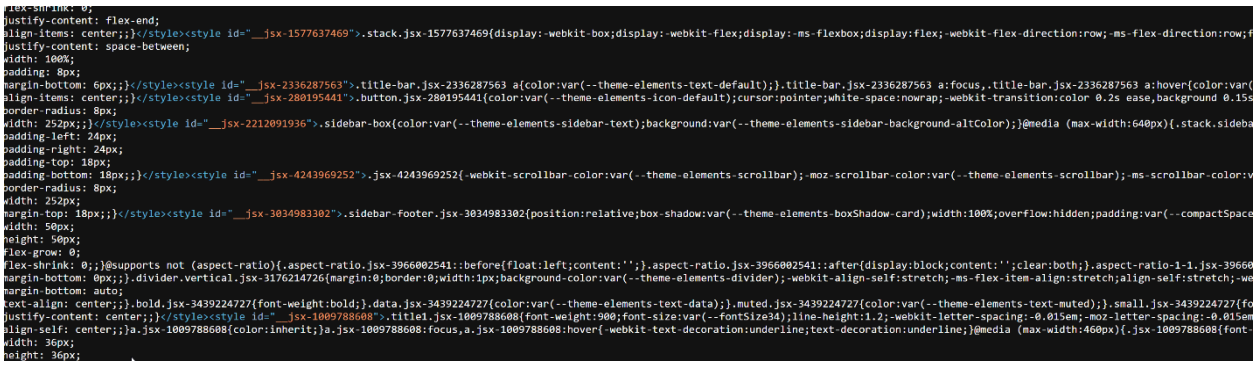


Рисунок 4.15 – Внутрішнє представлення сторінки IGN

Результати експерименту наведені на рис. 4.16 - 4.17.

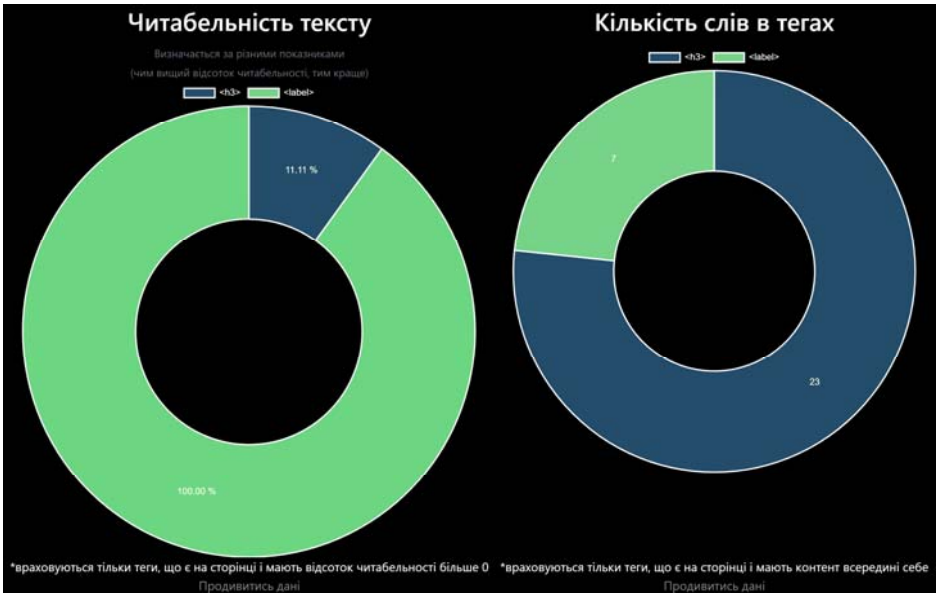


Рисунок 4.16 – Текстовий аналіз сторінки IGN

За текстовим графіком, можемо побачити, що виділено лише два елементи – заголовки третього рівня та підписи до елементів. Інші текстові елементи відсутні.

Таблиця 4.7 – Текстовий аналіз сторінки IGN

Назва елементу	Кількість слів	Читабельність, %
<h3>	23	11.11
<label>	7	100

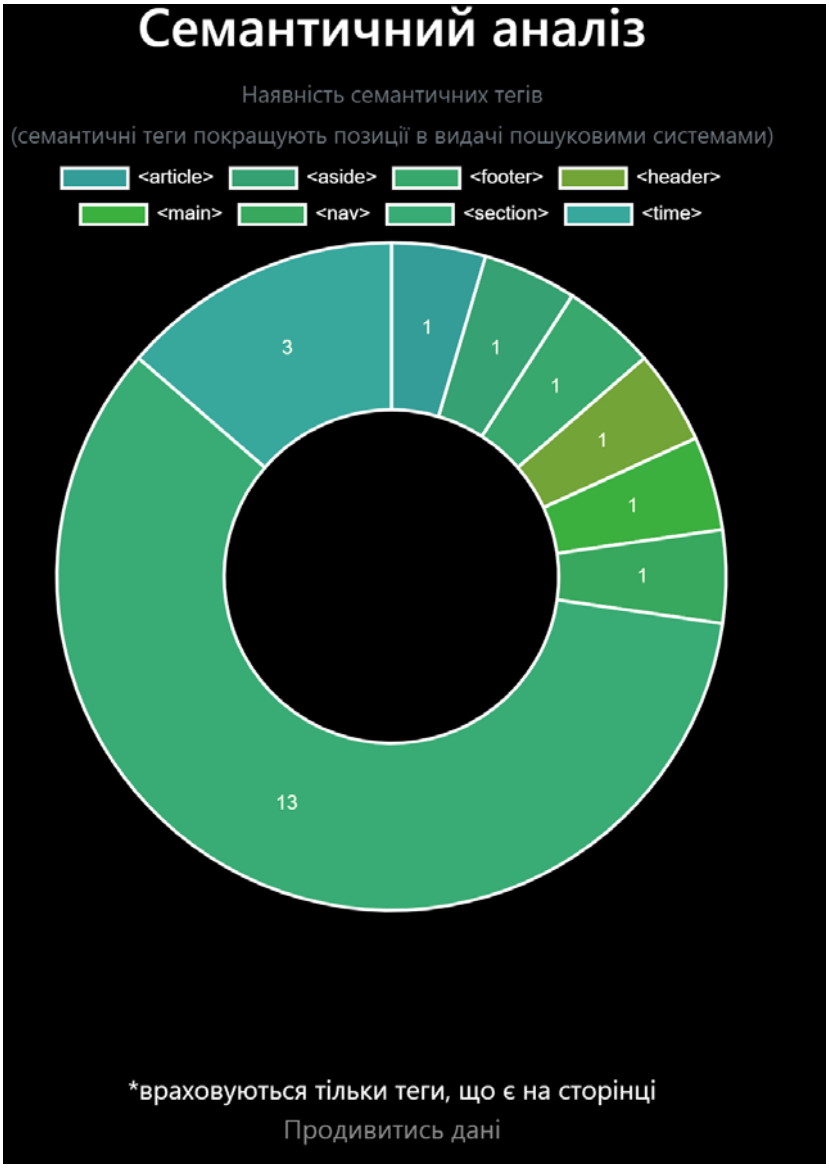


Рисунок 4.17 – Семантичний аналіз сторінки IGN

На графіку спостерігаємо велику кількість різноманітних семантичних тегів, секції розділення вмісту сторінки.

Таблиця 4.8 – Семантичний аналіз сторінки IGN

Назва елементу	Кількість
<article>	1
<aside>	1
<footer>	1
<header>	1
<main>	1
<nav>	1
<section>	13
<time>	3

Висновки: текстовий аналіз демонструє високу читабельність підписів до елементів списку, але невелике значення для заголовків. Для покращення читабельності, необхідно або замінити заголовки на інші текстові теги, або змінити їх структуру.

Семантичний аналіз демонструє високий рівень для аналізу сторінки пошуковими системами, що гарантує високу вірогідність індексації сторінки при пошукових запитах. Велика кількість семантичних тегів також покращує навігацію по вмісту веб-сайту для розробника.

4.2.5 Експеримент 5

Мета: Провести експеримент по аналізу інформаційної платформи Xbox Wire.

Вхідні дані – html файл головної сторінки веб-сайту, зображення якої наведено на рис. 4.18 та внутрішнє представлення на рис. 4.19.

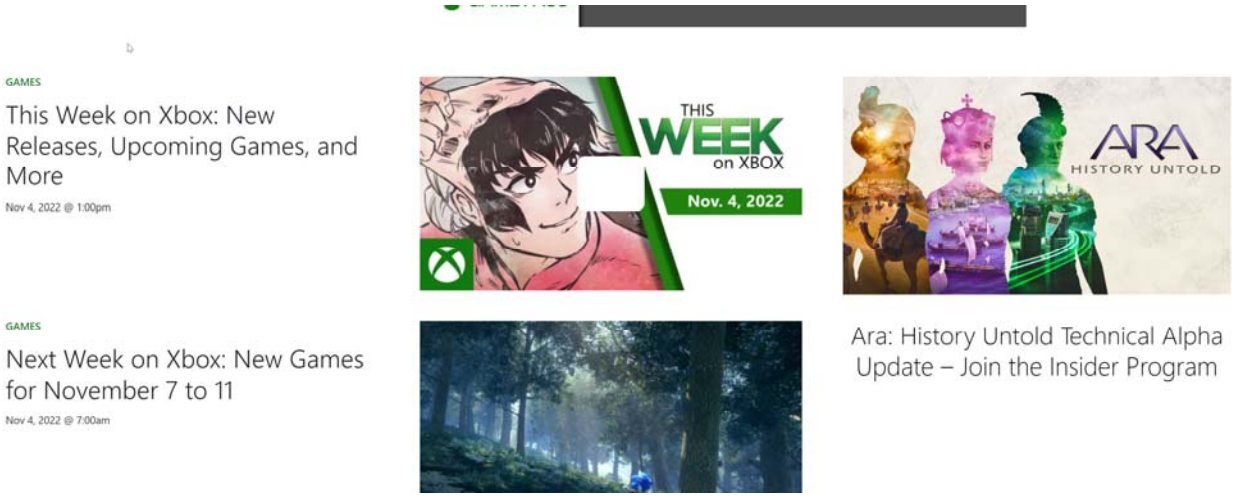


Рисунок 4.18 – Головна сторінка Xbox Wire

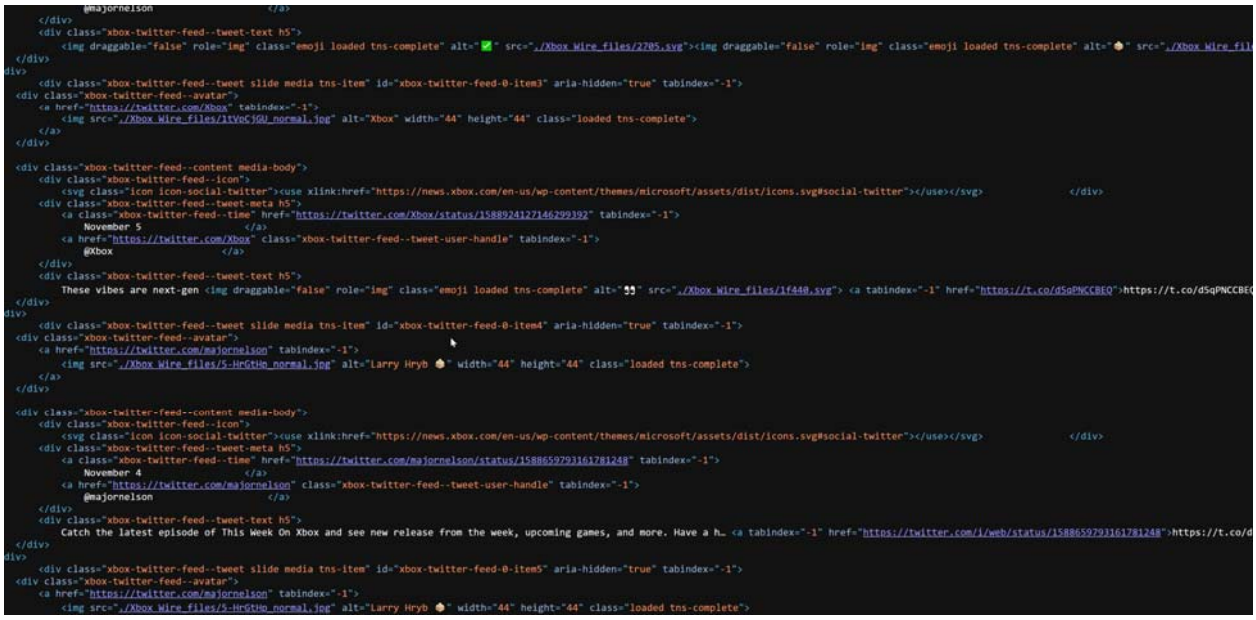


Рисунок 4.19 – Внутрішнє представлення Xbox Wire

Результати експерименту наведені на рисунках 4.20 - 4.21.

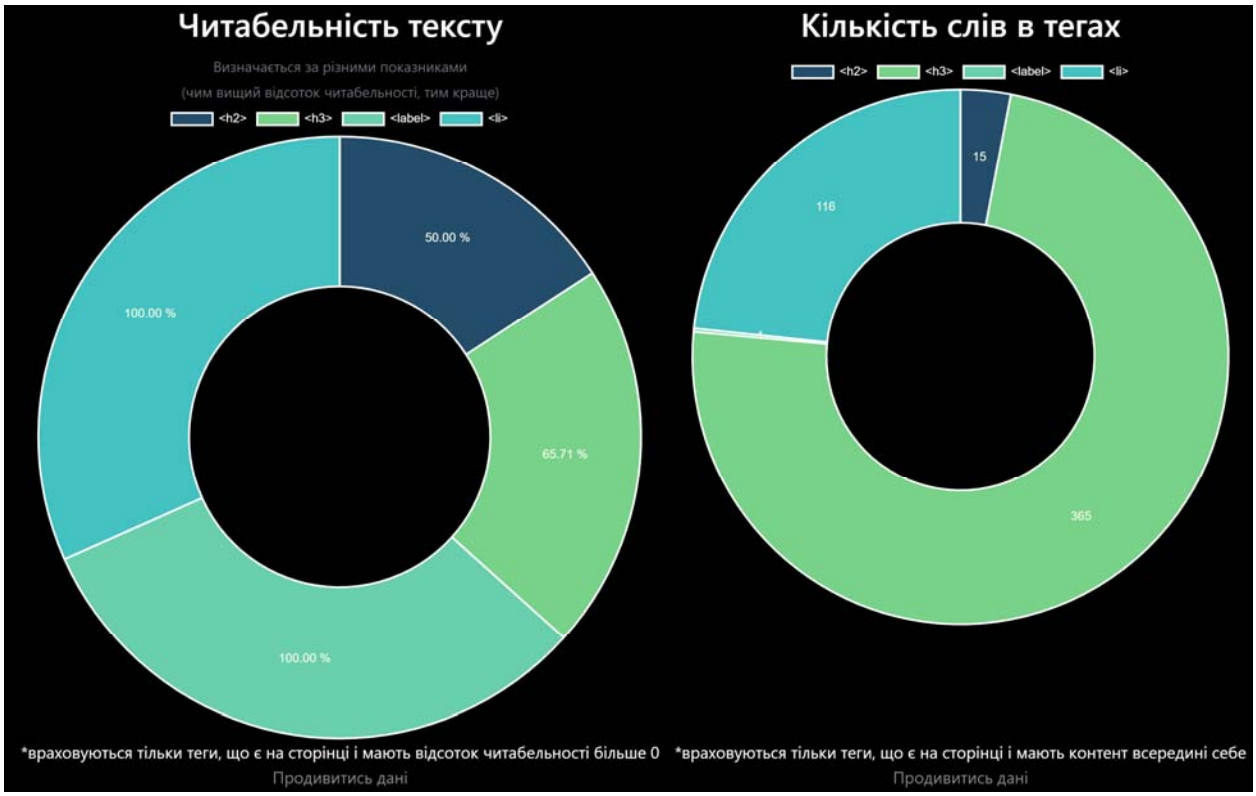


Рисунок 4.20 –Текстовий аналіз Xbox Wire

На графіках можна побачити, що більша частина тексту припадає на заголовки третього рівня та списки. Найвищий відсоток читабельності складають підписи до елементів та списки.

Таблиця 4.9 – Текстовий аналіз сторінки Xbox Wire

Назва елементу	Кількість слів	Читабельність, %
<h2>	15	50
<h3>	365	65.71
<label>	1	100
	116	100

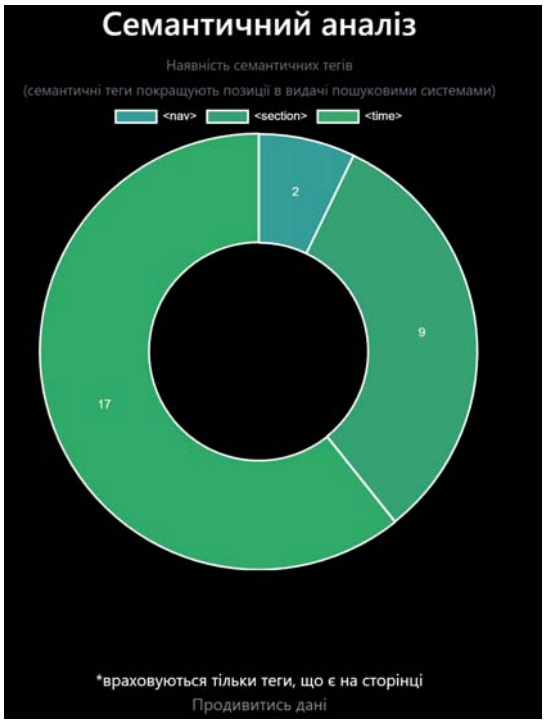


Рисунок 4.21 – Семантичний аналіз Xbox Wire

На графіку спостерігаємо невелику кількість семантичних тегів, зустрічаються в кількості більше одного.

Таблиця 4.10 – Семантичний аналіз сторінки Xbox Wire

Назва елементу	Кількість
<nav>	2
<section>	9
<time>	17

Висновки: текстовий аналіз демонструє високу читабельність для всіх елементів – всі елементи мають відсоток читабельності вищий або рівний 50%, що можна вважати результатом правильної структури тексту і гарантує легкість сприйняття та простоту читабельності тексту.

Семантичний аналіз має добре організовану організацію внутрішньої структури, завдяки використанню секцій, але є загальні проблеми з основними семантичними тегами – не має розділу на верхню та нижню частини веб-

сторінки, тег з навігацією зустрічається більше одного разу, що може викликати проблеми при пошуковій видачі сторінки пошуковими системами.

4.3 Висновки результатів експериментів

В ході проведення серії експериментів, було виявлено, що текстовий модуль контекстного дослідження дозволяє ефективно перевіряти проблемні теги на предмет читабельності, а також аналізувати кількість слів в тегах, для оптимізації в пошукових системах, що корисно для SEO-фахівців. Семантичний аналіз, для кількісної оцінки семантичних тегів, також продемонстрував свою ефективність – він дає можливість оцінити користувачу, наскільки добре сторінка використовує необхідні теги, для кращого позиціонування в пошукових системах, а для розробника – зрозуміти, чи добре налаштована структура сторінки, чи використовуються правильно семантичні теги за своїм призначенням, чи є наявними розділення структурних частин сторінки, для легшої навігації під час рефакторингу.

За результатами експерименту, візуалізували дані у вигляді згрупованих гістограм, представлені на рисунках 4.23 - 4.24.

Результати текстового аналізу, читабельності сторінки показали, що найбільший відсоток читабельності зосереджений серед підписів та елементів списку, серед всіх сторінок в експерименті. Ближчі до середніх результатів читабельності, мають заголовки третього рівня. Найменший відсоток читабельності зосереджений в строкових елементах, але цей елемент зосереджений лише в одній сторінці.

Семантичний аналіз, кількісного підрахунку семантичних тегів показав, що найбільш поширеними елементами стали елементи для представлення часу, розбиття сторінки на секції, а також основні теги для верхньої та нижньої секції. Найменшу частку складають теги для внутрішнього змісту сторінки, тег бокової панелі.

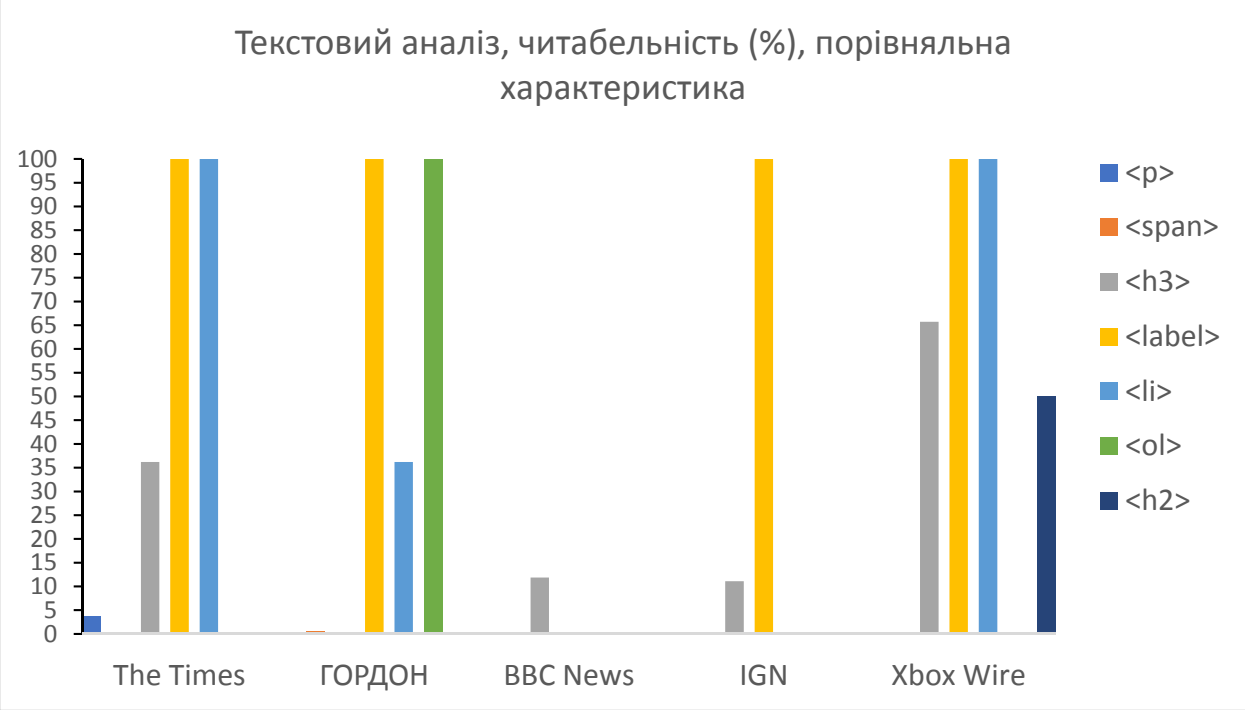


Рисунок 4.23 – Графік читабельності тегів

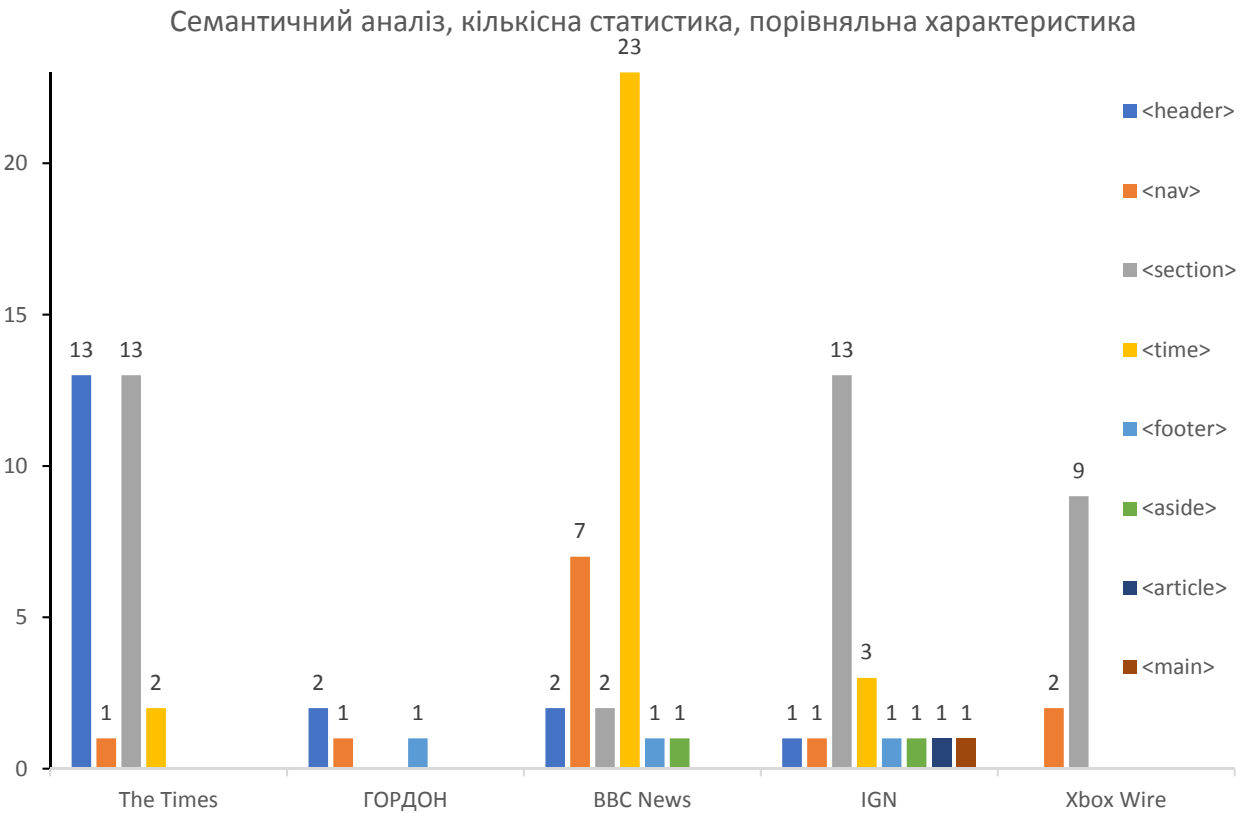


Рисунок 4.24 – Графік кількісного підрахунку семантичних тегів

ВИСНОВКИ

Під час аналізу предметної області, були виділені основні проблеми при аналізі та дослідження веб-сайті, проведене широкий аналіз проблем з посиланням на статистичні дані, а також дослідження компаній залучених до веб-розробки. Складено план та критерії майбутніх модулів з урахуванням методу контекстних досліджень, для розробки програмного рішення.

Проведений аналіз технологічної платформи, бібліотек, програмного середовища, представлений їх опис, а також обґрунтування вибору, при розробці програмного рішення.

Було розроблено програмний додаток, з поясненням етапів розробки, з представленням діаграм та схем, описані елементи системи, їх взаємодію та сценарії їх роботи, представлена логіка поведінки під час експлуатації на демонстраційних даних.

Етапи розробки та методологія проектування, дозволить розширювати функціонал модулів системи, вносити корективи у вже існуючі, розробляти нові.

Під час проведення експериментів, було що метод контекстних досліджень можливо використовувати під час аналізу веб-сайтів та їх окремих сторінок. Через глибину досліджуваної теми контекстного дослідження, неможливо охопити її всю – лише в тих критеріях, яку ми визначили в рамках експерименту, розробленими модулями. За рахунок наукової новизни та актуальності інформаційних технологій, пов’язаних з глобальною мережею, можливе подальше поглиблене вивчення практичного використання методу контекстних досліджень, із застосуванням нових критеріїв, для дослідження можливостей та доцільності їх практичного використання.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1 The Beginner's Guide to Website Development. URL: https://blog.hubspot.com/website/website-development (дата звернення 10.08.2022).

2 Number of Internet Users in 2022/2023: Statistics, Current Trends, and Predictions. URL: https://financesonline.com/number-of-internet-users/ (дата звернення 10.08.2022).

3 Modern web development: what makes it ‘modern’? URL: https://hub.packtpub.com/modern-web-development-what-makes-it-modern/ (дата звернення 10.08.2022).

4 HTTP requests. URL: https://www.ibm.com/docs/en/cics-ts/5.3?topic=protocol-http-requests (дата звернення 10.08.2022).

5 Top 10 BEST Browsers For PC [2022 Web Browser Ranking]. URL: https://www.softwaretestinghelp.com/best-browser-ranking/ (дата звернення 10.08.2022).

6 Oracle, databases. URL: https://www.oracle.com/database/what-is-database/ (дата звернення 10.08.2022).

7 The best code editors in 2022: our guide to the top options. URL: https://www.creativebloq.com/advice/best-code-editors (дата звернення 10.08.2022).

8 IDE vs Code Editor – Why and When to Use Them. URL: https://www.jobsity.com/blog/ide-vs-code-editor-why-and-when-to-use-them (дата звернення 10.08.2022).

9 Top Programming Languages and Web Frameworks To Learn in 2022. URL: https://medium.com/dare-to-be-better/top-programming-languages-and-web-frameworks-to-learn-in-2022-776113bfefeb (дата звернення 10.08.2022).

10 10 базовых принципов SEO продвижения. URL: https://tilda.education/articles-how-seo-works (дата звернення 10.08.2022).

11 SEO Specialist Job Description. URL: https://www.glassdoor.com/Job-Descriptions/SEO-Specialist.htm (дата звернення 10.08.2022).

12 Основные методы исследования Web-сайтов. URL: https://zoomgorod.ru/articles/osnovnyie-metodyi-issledovaniya-web-sajtov.html (дата звернення 10.08.2022).

13 Website Security: 3 Things to Consider When Sharing Confidential Information. URL: https://blog.quttera.com/post/3-things-to-consider-when-sharing-confidential-information/ (дата звернення 10.08.2022).

14 Що таке SSL-сертифікат. URL: https://hostiq.ua/wiki/ukr/ssl-certificate/ (дата звернення 10.08.2022).

15 User Interface Design In Modern Web Applications. URL: https://www.smashingmagazine.com/2011/09/user-interface-design-in-modern-web-applications/ (дата звернення 10.08.2022).

16 What Is Service-Oriented Architecture?. URL: https://medium.com/@SoftwareDevelopmentCommunity/what-is-service-oriented-architecture-fa894d11a7ec (дата звернення 10.08.2022).

17 CloudFire, What is site speed? URL: https://www.cloudflare.com/learning/performance/why-site-speed-matters/ (дата звернення 10.08.2022).

18 5 Challenges in Web Application Development. URL : https://marutitech.com/5-challenges-in-web-application-development/#Performance (дата звернення 10.08.2022).

19 How To Build Successful Websites For Your Clients (In 8 Steps). URL: https://www.a2hosting.com/blog/build-successful-client-website/ (дата звернення 10.08.2022).

20 How to Price a Website for a Client: A Comprehensive Guide. URL: https://power-plugins.com/features/how-to-price-a-website-for-a-client-a-comprehensive-guide/ (дата звернення 10.08.2022).

21 5 основных проблем веб-дизайна и разработки. URL: https://blog.depositphotos.com/ru/veb-dizajn-i-razrabotka.html (дата звернення 10.08.2022).

22 Decision Theory. What is Decision Theory?. URL: https://corporatefinanceinstitute.com/resources/wealth-management/decision-theory/ (дата звернення 10.08.2022).

23 Техническое задание на программу (ПО). URL: https://www.swrit.ru/tz-na-programmu.html (дата звернення 10.08.2022).

24 Разработка спецификаций. URL: https://evergreens.com.ua/ru/development-services/srs-development.html (дата звернення 10.08.2022).

25 Usability Researcher Job Description, Career as an Usability Researcher, Salary, Employment. URL: https://careers.stateuniversity.com/pages/141/Usability-Researcher.html (дата звернення 10.08.2022).

26 Карточная сортировка. URL: https://usabilitylab.ru/usability/kartochnaya-sortirovka/ (дата звернення 10.08.2022).

27 Prototyping. URL: https://www.usability.gov/how-to-and-tools/methods/prototyping.html (дата звернення 10.08.2022).

28 30+ Website usability survey questions to understand user behavior. URL: https://www.questionpro.com/blog/website-usability-survey-questions/ (дата звернення 11.08.2022).

29 Интервью и фокус-группы. URL: https://usabilitylab.ru/usability/intervyu-i-fokus-gruppyi/ (дата звернення 11.08.2022).

30 Contextual Inquiry: Inspire Design by Observing and Interviewing Users in Their Context. URL: https://www.nngroup.com/articles/contextual-inquiry/ (дата звернення 11.08.2022).

31 What is a website builder, and how does it work? URL: https://www.one.com/en/websitebuilder/what-is-a-website-builder-and-how-does-it-work (дата звернення 11.08.2022).

32 What Is The Data Cloud? URL: https://www.snowflake.com/trending/data-cloud-storage (дата звернення 11.08.2022).

33 VoIP (voice over Internet Protocol). URL: https://www.techtarget.com/searchunifiedcommunications/definition/VoIP (дата звернення 11.08.2022).

34 Techopedia Explains Abstract Class. URL: https://www.techopedia.com/definition/17408/abstract-class (дата звернення 11.08.2022).

35 Copywritely. URL: https://copywritely.com/ru/word-counter/ (дата звернення 11.08.2022).

36 Keyword Analysis: How to Analyze Your Keywords for SEO. URL: https://www.semrush.com/blog/keyword-analysis-how-to-evaluate-the-best-keywords/ (дата звернення 11.08.2022).

37 Website Words Counter. URL: https://sitechecker.pro/ru/website-word-counter/ (дата звернення 11.08.2022).

38 Why is website readability important? URL: https://www.hotjar.com/conversion-rate-optimization/glossary/website-readability/ (дата звернення 11.08.2022).

39 What is Website Readability And How Do I Improve It? URL: https://www.bkacontent.com/understanding-readability-when-you-buy-website-content/ (дата звернення 11.08.2022).

40 Website Font Size: Size does matter. URL: https://www.liveseysolar.com/website-font-size-size-does-matter/#:~:text=Larger%20website%20font%20size%20improves,and%20result%20in%20improved%20usability.%E2%80%9D (дата звернення 11.08.2022).

41 The Problem With Long-Scroll, Single Page Websites. URL: https://www.walkersands.com/the-problem-with-long-scroll-single-page-websites/ (дата звернення 11.08.2022).

42 The Readability Formula: Making Your Website Easy-to-Read. URL: https://kickpoint.ca/the-readability-formula-making-your-website-easy-to-read (дата звернення 11.08.2022).

43 HTML Semantic Elements. URL: https://www.w3schools.com/html/html5_semantic_elements.asp (дата звернення 11.08.2022).

44 How To Do Semantic Analysis In SEO? URL: https://www.twaino.com/en/seo/semantic-analysis/ (дата звернення 11.08.2022).

45 Research: Definition, Characteristics, Goals, Approaches. URL: https://www.iedunote.com/research-definition-characteristics-goals-approaches (дата звернення 11.08.2022).

46 What Is Empirical Research? Definition, Types & Samples. URL: https://research.com/research/what-is-empirical-research (дата звернення 12.08.2022).

47 Deductive Approaches to Research. URL: https://pressbooks.bccampus.ca/jibcresearchmethods/chapter/1-7-deductive-approaches-to-research/ (дата звернення 12.08.2022).

48 Empirical Research: Definition, Methods, Types and Examples. URL: https://www.questionpro.com/blog/empirical-research/ (дата звернення 12.08.2022).

49 What is PhpStorm? URL: https://www.javatpoint.com/phpstorm (дата звернення 12.08.2022).

50 Advantages and Disadvantages of JavaScript. URL: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-javascript/ (дата звернення 12.08.2022).

51 4 reasons why JavaScript is so popular. URL: https://opensource.com/article/20/11/javascript-popular (дата звернення 12.08.2022).

52 Importance of SCSS. URL: https://codeit.mk/home/followUs/blog/IMPORTANCE-OF-SCSS.html (дата звернення 12.08.2022).

53 GitHub Docs. URL: https://docs.github.com/en/get-started/quickstart/hello-world (дата звернення 12.08.2022).

54 How To Configure SSH Key-Based Authentication on a Linux Server. URL: https://www.digitalocean.com/community/tutorials/how-to-configure-ssh-key-based-authentication-on-a-linux-server (дата звернення 12.08.2022).

55 Embedded JavaScript templating. URL: https://ejs.co/ (дата звернення 12.08.2022).

56 What is Node.js: A Comprehensive Guide. URL: https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-nodejs (дата звернення 12.08.2022).

57 Event Loops in NodeJS – Beginner's Guide to Synchronous and Asynchronous Code. URL: https://www.freecodecamp.org/news/nodejs-eventloop-tutorial/ (дата звернення 13.08.2022).

58 What Are The Benefits Of Using Express.Js For Backend Development. URL: https://www.techomoro.com/what-are-the-benefits-of-using-express-js-for-backend-development/ (дата звернення 13.08.2022).

59 Gulp Overview. URL: https://www.tutorialspoint.com/gulp/gulp_overview.htm (дата звернення 13.08.2022).

60 Nodemon. URL: https://www.npmjs.com/package//nodemon (дата звернення 13.08.2022).

61 Sass Documentions. URL: https://sass-lang.com/documentation/ (дата звернення 13.08.2022).

62 Bootstrap. URL: https://getbootstrap.com/ (дата звернення 13.08.2022).

63 Bootstrap Grid. URL: https://mdbootstrap.com/docs/standard/layout/grid/ (дата звернення 13.08.2022).

64 Chart.js. URL: https://www.w3schools.com/js/js_graphics_chartjs.asp (дата звернення 13.08.2022).

65 Use EJS as Template Engine in Node.js. URL: https://www.geeksforgeeks.org/use-ejs-as-template-engine-in-node-js/ (дата звернення 13.08.2022).

66 jQuery. URL: https://jquery.com/ (дата звернення 13.08.2022).

67 jsdom. URL: https://www.npmjs.com/package/jsdom (дата звернення 13.08.2022).

68 SOLID – Single responsibility. URL: https://stackify.com/solid-design-principles/ (дата звернення 13.08.2022).

69 Unified Modeling Language (UML), Activity Diagrams. URL: https://www.geeksforgeeks.org/unified-modeling-language-uml-activity-diagrams/ (дата звернення 13.08.2022).

70 Working with JSON. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON (дата звернення 13.08.2022).

ДОДАТКИ

ДОДАТОК А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ

Проректор
Українського державного університету науки і технології
Анатолій РАДКЕВИЧ

КОНТЕКСТНЕ ДОСЛІДЖЕННЯ WEB-САЙТУ

Технічне завдання
ЛИСТ ЗАТВЕРДЖЕННЯ
1116130.01257-01-ЛЗ

Завідувач кафедри КІТ

_____Вадим ГОРЯЧКІН

Керівник розробки

_____Олександра ГОРБОВА

Виконавець

_____Данило БОГУЦЬКИЙ

Нормоконтролер

_____Світлана ВОЛКОВА

2022

ЗАТВЕРДЖЕНО

1116130.01257-01

КОНТЕКСТНЕ ДОСЛІДЖЕННЯ WEB-САЙТУ

Технічне завдання

Листів 25

2022

АНОТАЦІЯ

Документ 1116130.01257-01 «Контекстне дослідження Web-сайту» входить до складу програмної документації на програму, яка виконує аналіз веб-сторінки, за допомогою методу контекстного дослідження.

Об’єкт дослідження – гіпертекстові документу за визначеними критеріями, з урахуванням методу контекстних досліджень.

Для вирішення задачі, використовується метод контекстних дослідження, який включає в себе детальне дослідження та інтерв’ю невеликої групи людей за темами предметної області, для отримання реального представлення досліджуваного веб-сайту.

Програма являється веб-додатком, з використанням наступних технологій: HTML, CSS, JavaScript, PHP, Apache2. Конфігурація комп'ютера стандартна. Комплекс функціонує як в середовищі MS WINDOWS, так і на сімействі систем Unix.

В даному документі вказано призначення і область застосування програмного продукту, технічні, техніко-економічні показники, що пред'являються до програми, терміни розробки проекту.

ЗМІСТ

ВСТУП

Контекстне дослідження Web-сайту – програмний продукт, у вигляді веб-додатку, для аналізу гіпертекстових документів за визначеними критеріями, з урахуванням методу контекстних досліджень.

Необхідність розробки програмного застосунку зумовлена відсутністю аналогів на ринку.

Область застосування програмного застосунку – глобальна мережа Інтернет. Основна аудиторія – користувачі, що мають власний веб-сайт – готовий чи який знаходиться на етапі розробки.

1 ПІДСТАВА ДЛЯ РОЗРОБКИ

Основою для розробки є наказ ректора Українського державного університету науки і технології Радкевич А.В. «Про затвердження тем та призначення керівників дипломних проектів» №01206 ст. від 10.12.2022 р.

Тема проекту: «Контекстне дослідження Web-сайту».

Керівник дипломного проекту: доцент кафедри «КІТ» О.В. Горбова.

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

2.1 Функціональне призначення

Основним функціональним призначенням програмного додатку є робота з гіпертекстовими документами, за визначеним набором критерій, для аналізу даних вмісту сторінки, з урахуванням методу контекстних досліджень.

Програма, що розробляється, повинна надавати зручний призначений для користувача інтерфейс і виконувати наступні функції:

1) можливість передавати гіпертекстові файли;
2) проводити аналіз завантажених гіпертекстових документів, відповідно до заданих критеріїв;
3) виводити результат аналізу гіпертекстових документів;
4) можливість графічного та текстового представлення програми.

2.2 Експлуатаційне призначення

Основним експлуатаційним напрямом програми є вдосконалення якості створюваного продукту, спрощення взаємодії користувача та розробника та установка точних цілей для розробника стосовно виконуваних задач, які формуються на основі потреб користувача.

Мета контекстних досліджень полягає в:

– удосконаленні збору інформації на стадії створення концепту;
– удосконалення фільтрації зібраної інформації, з урахуванням контексту;
– налаштовуванні спільної взаємодії між можливістю та способу технічної реалізації виконавців відповідно до функціональних вимог користувачів;
– удосконалення ефективності роботи з точки зору витраченого часу на розробку та людського ресурсу.

3 ВИМОГИ ДО ПРОГРАМИ

3.1 Вимоги до функціональних характеристик

Програма повинна виконувати наступні функції:

1) надавати можливість передавати гіпертекстові документи, для подальшого завантаження на веб-сервер, для їх аналізу;

2) проводити аналіз за визначеними програмою критеріями, з урахуванням контекстного дослідження;

3) надавати текстове, графічне або схематичне представлення аналізу по кожному із завантажених гіпертекстових документів.

3.2 Введення і формування даних

Введення початкових даних здійснюється в діалоговому режимі, з повідомленням системи, якщо дані були введені некоректно.

3.3 Вимоги до вхідних даних

До початкових даних належить гіпертекстовий документ.

Гіпертекстовий документ має бути представлений у форматі .html або .htm та відповідати сучасному і актуальному стандарту веб-документів HTML5. Допускається наявність в документі скриптів JS, правил для каскадних таблиць стилей CSS. Текст документу необмежений, має відповідати кодуванню UTF-8, з підтримкою Unicode.

Введення початкових даних здійснюється за допомогою графічного інтерфейсу програмного меню.

Гіпертекстовий документ може бути завантажений в окремому компоненті у вигляді файлу. Фільтр для компоненту – тільки файли з форматами .html та .htm. Пошук та додавання використовується згідно системному провіднику файлів, в залежності від використаної системи. Виконується за допомогою клавіатури та миші.

Набір критеріїв для аналізу є сталим та влаштованим в програму. Назва, порядок представлення, принцип дії та логіка визначена розробником, зберігається в форматі модулів.

3.2 Вимоги до надійності

Одним з критеріїв правильного функціонування програмного продукту є забезпечення надійності програмного продукту. Надійність програмного продукту передбачається за рахунок таких умов і засобів:

– програма повинна бути стійка до вхідних даних;

– наявність архівної копії тексту програми на зовнішньому носії;

– кількість відмов програми визначається не більше однієї відмови на 100 запусків.

3.3 Умови експлуатації

Даний програмний продукт може бути застосований в будь-яких установах і приміщеннях, призначених для роботи з ЕОМ і відповідних вимогам по пожежо- та електро-безпеки.

Умови для експлуатації програмного продукту повинні бути схожими з умовами на обчислювальному центрі (температура повітря 21-25°С, відносна вогкість 60±15%, атмосферний тиск 630-800 мм.рт.ст.).

Для обслуговування програмного продукту достатньо одного програміста. Роботу з програмою може виконувати людина яка має досвід роботи з персональним комп’ютером та ознайомлена з керівництвом користувача.

3.4 Вимоги до складу і параметрів технічних засобів

Мінімальна конфігурація комп'ютера для забезпечення працездатності програмного продукту:

– Intel Pentium IV з частотою процесора від 2.0 Ггц;

– ОЗУ 256 Мб;

– 20 Гб на жорсткому диску;

– дисковод 3.5", 1.4 Мбайт;

– CD-ROM;

– дисплей VGA;

– принтер струменевий / лазерний;

– стандартна клавіатура (104/105 клавіші);

– миша.

3.5 Вимоги до інформаційної і програмної сумісності

Для функціонування програмного продукту необхідна наявність встановлених:

1) операційної системи Windows 98/NT/2000/XP або системи UNIX;

2) наявність пакету LAMP, для UNIX або набору OpenServer для Windows.

3.6 Вимоги до маркування і упаковки

Приклад маркування приведений на рис. 3.1.

Програмне забезпечення
«Контекстне дослідження Web-сайту»
УДУНТ. Кафедра КІТ.
Версія 1.0, 2022 р.
Розробник: Богуцький Д.

Рисунок 3.1 – Приклад маркування

3.7 Вимоги до транспортування і зберігання

Копія програмного продукту повинна зберігатися на жорсткому диску. Вимоги до зберігання повинні відповідати вимогам по експлуатації гнучких/жорстких дисків.

Транспортування програмного продукту може здійснюватися за допомогою CD-дисків.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Програмна документація на дану програму повинна включати:

– специфікація;

– текст програми;

– опис програми;

– керівництво користувача.

Вся документація до програми повинна задовольняти вимогам державного стандарту до оформлення програмних документів [9].

5 ТЕХНІКО–ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ ПРОЕКТУ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ

Мета такого обґрунтування – провести оцінку витрат та мінімізувати їх. Окрім цього, необхідно провести оцінку вартості кінцевого продукту та процесу розробки.

Початковим етапом розрахунків – оцінка розміру програмного забезпечення.

Згідно моделі COCOMO, розмір проекту S вимірюється в рядках коду LOC / KLOC, а трудовитрати в людино-місяцях,

$$E = a \times S^b \times EAF \quad (5.1)$$

де E – витрати праці на проект;

S^b – обсяг коду;

EAF – фактор уточнення витрат.

Розмір програмного коду було підраховано за допомогою інструменту cloc – інструмент, який підраховує порожні рядки, рядки коментарів та фізичні рядки вихідного коду в багатьох мовах програмування [1]. Загальний обсяг програмного коду становить 739 рядків, див. рис. 5.1.

Рисунок 5.1 – Підрахунок розміру програмного коду

$$E = 2{,}4 \times 0{,}739^{1{,}05} \times 1 = 1{,}77 \qquad (5.1)$$

Отже, згідно моделі COCOMO, орієнтовні трудовитрати на проект складуть приблизно 1,77 людино-місяці.

Нижче наведені розрахунки вартості розробки програмного забезпечення «Контекстне дослідження Web-сайту». Основними витратами визначено такі критерії як заробітна плата, відрахування на соціальні потреби, накладні та додаткові витрати, витрати на персональний комп'ютер на програмні засоби.

Розрахуємо заробітну плату інженера-програміста, для оцінки основної заробітної плати. Згідно статистиці сайту вакансій Work.ua, середня зарплатня інженера-програміста на 2022 рік, становить близько 17500 гривень – мінімальна 6000-9000 гривень, максимальне значення – близько 25000-35000 гривень [2]. Графік заробітної плати, див. рис. 5.2.

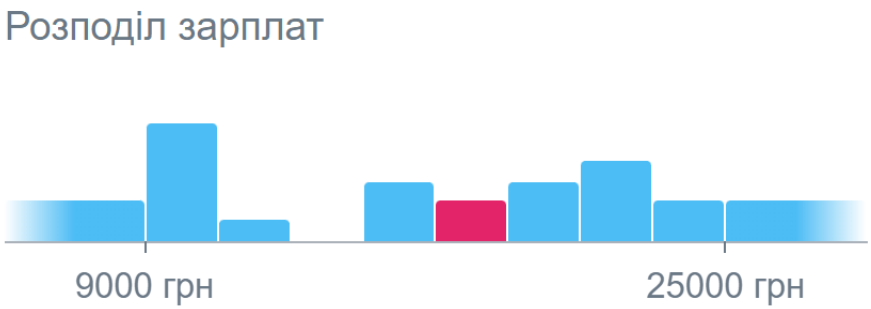


Рисунок 5.2 – Розподіл заробітної плати за Work.ua

Розрахунок заробітної платні проводиться по формі табл. 5.1.

Таблиця 5.1 – Фонд місячної заробітної плати

№ п/п	Посада виконавця	Оклад, грн/міс	Кількість		Сума зарплати грн
			чол	місяців	

1	інженер-програміст	17500	1	1,77	30975

Описаний в проекті програмний додаток був розроблений одним програмістом в період з 27.09.22 до 22.11.22, що становить 40 днів або приблизно 8 робочих тижнів. Витрати робочого часу прийняті за 40 годин у тиждень. Погодинна ставка кваліфікованого інженера–програміста складає 73 грн/год, згідно середній зарплаті (в середньому, робочий день інженера-програміста складає 8 годин). Таким чином, витрачено робочого часу:

$$t_{розробки} = N_{чол} \times N_{тиж} \times N_{год} \quad (5.2)$$

де $N_{чол}$ – кількість виконавців, чол;

$N_{тиж}$ – тривалість розробки;

$N_{год}$ – витрати робочого часу, год;

$$t_{розробки} = 1 \times 8 \times 40 = 320 \text{ чол/год} \quad (5.2)$$

ОЗП визначається за формулою:

$$ОЗП = t_{розробки} \times N \times ККВ \quad (5.3)$$

де $t_{розробки}$ – витрати праці у чол/год;

N – погодинна ставка;

ККВ – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. Коефіцієнт кваліфікації розробника (k) – ступінь підготовленості виконавця до дорученої йому роботі (він визначається

залежність від стажу праці та становить:

- для працюючих до 2 років- 0,75;
- від 2 до 3 років 1,0;
- від 3 до 5 років - 1,1-1,2;
- від 5 до 7 років - 1,3-1,4;
- понад 7 років - 1,5-1,6.

В даному випадку KKB приймається 0,75. ОЗП складає:

$$ОЗП = 320 \times 73 \times 0{,}75 = 17520 \quad (5.3)$$

Єдиний соціальний внесок на 2022 рік складає 22% [4]. Відрахування на соціальні потреби встановлюються у відсотках від суми заробітної плати:

$$С_{соц} = \frac{ОЗП \times 22\%}{100\%} \quad (5.4)$$

$$С_{соц} = \frac{17520 \times 22\%}{100\%} = 3854 \text{ грн.} \quad (5.4)$$

Отримані результати підсумовуються. Вони складають 21374 грн. та визначають основні прямі витрати.

Накладні витрати враховують такі витрати для роботи над проектом як: опалення, електроенергія, амортизація будівель, зарплату адміністративного персоналу і т.д.

$$С_{накл} = \frac{(ОЗП + С_{соц}) \times 35\%}{100\%} \quad (5.5)$$

$$С_{накл} = \frac{(17520 + 3854) \times 35\%}{100\%} = 7480 \text{ грн.} \quad (5.5)$$

Протягом усього терміну використання нової техніки підприємство щорічно витрачає певні кошти, пов’язані з її експлуатацією.

Експлуатаційні витрати на персональний комп’ютер визначаються протягом терміну розробки програмного засобу в залежності від вартості комп’ютеру. В експлуатаційні витрати входять:

– вартість витратних матеріалів;

– витрати на ремонт;

– заробітна плата ремонтника;

– оренда приміщення;

– додаткові витрати – прибирання приміщення, охорона, оренда, комунальні послуги;

– амортизаційні витрати на персональний комп'ютер і програмне забезпечення;

– витрати на електроенергію ($С_{ел}$), які визначаються за формулою:

$$С_{ел} = Р \times В \times Т_{розр} \quad (5.6)$$

- Р – потужність комп’ютера та допоміжних споживачів електричної енергії, приймається 0,45 кВт/год;
- В – вартість 1 кВт/годин для непобутових споживачів, становить близько 1,68 грн [4];
- $Т_{розр}$ – час роботи з ЕВМ, приймається рівним робочому часу.

Витрати на електроенергію визначаються так:

$$С_{ел} = 0{,}45 \times 1{,}68 \times 320 = \ 242 \text{ грн.} \quad (5.6)$$

Витрати на витратні матеріали ($С_{вм}$) протягом всього терміну експлуатації приблизно 10% від вартості комп'ютеру. Вартість робочої станції приймається 22 000 грн. [5], термін експлуатації – 5 років, відповідно до Податкового кодексу України [6]. Отже, можна визначити ці витрати за період створення програмного засобу:

$$С_{вм} = В_{ком} \times \frac{N_Д}{N_{експ} \times 365} \times \frac{10\%}{100\%} \quad (5.7)$$

де $B_{КОМ}$ – вартість персонального комп'ютеру;

$N_Д$ – кількість днів розробки програмного продукту;

$N_{експ}$ – термін експлуатації персонального комп'ютеру.

Витрати на витратні матеріали визначаються так:

$$С_{вм} = 22000 \times \frac{40}{5 \times 365} \times \frac{10\%}{100\%} = 48 \text{ грн.} \quad (5.7)$$

Заробітна плата ремонтника ($C_{рем}$) визначена наступним чином: на ремонт 50 комп'ютерів потрібен один інженер-системотехнік. За даними порталу Trud, середньомісячна заробітна плата інженера-системотехніка в Україні на 2022 рік становить 7000 грн [7], див. рис. 5.3.

Рисунок 5.3 – Статистика зарплати інженера-системотехніка Trud

Тоді в перерахунку на один комп'ютер його заробітна плата за період розробки програмного продукту складає:

$$C_{рем} = \frac{C_{рем}^{'}}{N_{ком}} \cdot T_{міс} \quad (5.8)$$

де $C_{рем}^{'}$ – середньомісячна заробітна плата;

$N_{КОМ}$ – кількість комп'ютерів на одного ремонтника.

$T_{мес}$ – час розробки програмного продукту, міс.

Заробітна плата ремонтника ($C_{рем}$) буде складати:

$$C_{рем} = \frac{70'00}{50} \cdot 1{,}77 = 248 \text{ грн.} \quad (5.8)$$

За статистикою витрати на комплектуючи вироби (СКОМ) для ремонту персонального комп'ютера складає 10% від його вартості за термін його експлуатації, тобто рівні витратам на витратні матеріали:

$$\text{СКОМ} = \text{СВМ} = 40 \text{ грн.} \quad (5.9)$$

Амортизаційні відрахування на персональний комп’ютер визначені з положення, що амортизаційний період в даний час дорівнює терміну морального старіння обчислювальної техніки і складає приблизно 3 роки, що являється терміном корисного використання техніки [8].

$$\text{АКП} = \text{В}_{\text{КОМ}} \times \frac{\text{N}_{\text{Д}}}{\text{N}_{\text{експ}} \times 365} \quad (5.10)$$

$$\text{АКП} = 22000 \times \frac{1{,}77}{3 \times 12} = 1082 \text{ грн.} \quad (5.10)$$

Для функціонування персонального комп’ютера використовувалася операційна система Windows 10 Home Edition, для написання програмного забезпечення – програмне середовище PHPStorm 2019, 30 днів безкоштовного використання, тому приймаємо його значення за 0 [9].

$$\text{АКП} = 3290 \times \frac{1{,}77}{5 \times 12} = 97 \text{ грн.} \quad (5.10)$$

$$\text{АКП} = 0 \times \frac{1{,}77}{5 \times 12} = 0 \text{ грн.} \quad (5.10)$$

Розрахунок амортизаційних відрахувань на програмне забезпечення зведений в табл. 5.2. Додаткові витрати ($C_{\text{ДОД}}$): прибирання приміщень, охорона, комунальні послуги важко оцінити точно і прийняти рівними 50% заробітної плати інженера-програміста, тобто 8750 гривень на місяць.

Оренду приміщень для однієї людини приймемо рівною 3000 гривень на місяць, з квадратурою 20м2 та 50м2 [10]. Тобто за весь період розробки (1,77) – 5310 грн. Сумарні експлуатаційні витрати на один персональний комп’ютер

складають:

$$С_{експ} = С_{ел} + С_{ВМ} + С_{рем} + АКП + АПО + С_{ор} + С_{дод} \quad (5.11)$$

$$С_{експ} = 242 + 40 + 248 + 1082 + 97 + 5310 + 8750 = 15769 \text{ грн.} \quad (5.11)$$

Результати розрахунків зведено у табл. 5.2 - 5.3.

Таблиця 5.2 – Використовуване програмне забезпечення

Найменування програмного забезпечення	Вартість програмного забезпечення, грн	Джерело придбання	Амортизаційні відрахування, грн
Windows 10	3290	https://soft.rozetka.com.ua/193810590/p193810590/	97
PHPStorm 2019	0	https://www.jetbrains.com/ru-ru/phpstorm/download/#section=windows	0
Всього:	3290		97

Таблиця 5.3 – Експлуатаційні витрати на ПК і ПЗ.

Найменування витрат	Витрати, грн
Витрати на електроенергію	242
Вартість витратних матеріалів	40
Витрати на ремонт	248
Амортизація персонального комп’ютера	1082
Амортизація програмного забезпечення	97

Оренда приміщення	5310
Додаткові витрати	8750
Всього	15769

Таким чином, витрати на створення програмного продукту складають:

$$С_{розробки} = ОЗП + С_{соц} + С_{накл} + С_{експ} \quad (5.12)$$

$$С_{розробки} = 17520 + 3854 + 7480 + 15769 = 44623 \text{ грн.} (5.12)$$

Розрахунок витрат зведено у табл. 5.4.

Таблиця 5.4 – Кошторис витрат на розробку програмного засобу

Найменування витрат	Витрати, грн
Основна заробітна плата	
Відрахування на соціальні потреби	
Накладні витрати	
Експлуатаційні витрати	
Всього	

За отриманими значеннями техніко-економічних показників проекту складено кошторис витрат на розробку сучасного програмного забезпечення для оцінки схожості програм. За результатами розрахунків, приблизна вартість розробки складає 44623 грн.

6 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Стадії і етапи розробки програмного продукту наведені в табл. 6.1

Таблиця 6.1 – Стадії і етапи розробки

Стадії розробки	Етапи робіт	Зміст робіт	Терміни
1. Технічне завдання	Обгрунтовування необхідності розробки програми	Постановка задачі. Збір висхідних матеріалів. Визначення структури вхідних і вихідних даних. Визначення вимог до технічних засобів. Вибір мови програмування.	25.02.2022 - 03.03.2022
	Розробка і затвердження технічного завдання	Визначення вимог до програми. Розробка техніко-економічного обгрунтовування розробки програми. Визначення стадій, етапів, термінів розробки програми і документації на неї. Узгодження і затвердження технічного завдання.	04.03.2022 - 12.03.2022 13.03.2022 - 26.02.2022

2. Робочий проект	Розробка проекту і програми	Розробка структур даних. Розробка специфікацій. Розробка призначеного для користувача інтерфейсу. Розробка програми. Тестування і відлагодження програми.	20.03.2022 - 20.03.2022 24.03.2022 - 24.04.2022 29.04.2022
		Розробка, узгодження і затвердження програмних документів.	25.05.2022 – 12.06.2022

7 ПОРЯДОК ВИКОНАННЯ І ПРИЙМАННЯ

Контроль здійснюється за допомогою виконання набору тестів з метою виявлення помилок в програмному продукті та його специфікаціях. Контроль за виконанням роботи здійснює головним керівником з розробки. Прийом здійснюється державною комісією.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Clock by AlDanial, GitHub Page. URL: https://github.com/AlDanial/cloc (дата звернення 10.11.2022).

2. Work.ua, зарплатня інженера-програміста. URL: https://www.work.ua/salary-%D0%B8%D0%BD%D0%B6%D0%B5%D0%BD%D0%B5%D1%80-%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%81%D1%82/?setlp=ua& (дата звернення 10.11.2022).

3. Єдиний соціальний внесок на 2022 рік. URL: https://services.dtkt.ua/catalogues/indexes/13 (дата звернення 10.11.2022).

4. Тариф на електроенергію на 2022 рік. URL: https://zn.ua/ECONOMICS/tarif-na-elektroenerhiju-skolko-budut-platit-ukraintsy-za-svet-v-2022-hodu.html (дата звернення 10.11.2022).

5. Comfy, робоча станція Asus X515JA-BR3853W. URL: https://comfy.ua/ua/noutbuk-asus-x515ja-br3853w-grey.html (дата звернення 10.11.2022).

6. Мінімально допустимі строки амортизації основних засобів та необоротних активів. URL: https://uteka.ua/ua/publication/news-14-ezhednevnyj-buxgalterskij-obzor-39-minimalno-dopustimye-sroki-amortizacii-osnovnyx-sredstv-i-vneoborotnyx-aktivov (дата звернення 10.11.2022)

7. Trud, зарплатня інженера-системотехніка. URL: https://ua.trud.com/ua/salary/2/67271.html (дата звернення 10.11.2022).

8. Термін корисного використання. URL: https://warez-portal.kiev.ua/termin-korisnogo-vikoristannya-orgtexniki (дата звернення 10.11.2022).

9. PHPStorm, trial version download. URL: https://www.jetbrains.com/ru-ru/phpstorm/download/#section=windows (дата звернення 10.11.2022).

10. Оренда приміщення. URL: https://www.olx.ua/d/obyavlenie/ofisnye-pomeschenie-20-m2-50m2-pr-gimnazicheskiy-IDQgiTK.html (дата звернення 10.11.2022).

11. Івченко, Ю.М. Основи стандартизації програмних систем [Текст]: методичні вказівки до дипломного проектування та лабораторних робіт / уклад.: Ю. М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна, 2009. - 38 с.

ДОДАТОК Б

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор
Українського державного
університету науки і технології
Анатолій РАДКЕВИЧ

КОНТЕКСТНЕ ДОСЛІДЖЕННЯ WEB-САЙТУ

Робочий проект
ЛИСТ ЗАТВЕРДЖЕННЯ
1116130.01257-01-ЛЗ

Завідувач кафедри КІТ
_____Вадим ГОРЯЧКІН
Керівник розробки
_____Олександра ГОРБОВА
Виконавець
_____Данило БОГУЦЬКИЙ
Нормоконтролер
_____Світлана ВОЛКОВА

2022

ЗАТВЕРДЖЕНО
1116130.01257-01

ДОДАТОК «КОНТЕКСТНЕ ДОСЛІДЖЕННЯ WEB-САЙТУ»

Специфікація

Листів 2

2022

Позначення	Найменування	Примітка
1116130.01257-01-ЛЗ 1116130.01257-01 12 01-ЛЗ 1116130.01257-01 12 01 1116130.01257-01 13 01-ЛЗ 1116130.01257-01 13 01 1116130.01257-01 ІЗ 01-ЛЗ 1116130.01257-01 ІЗ 01	Лист затвердження Лист затвердження Текст програми Лист затвердження Опис програми Лист затвердження Керівництво користувача	

ЗАТВЕРДЖЕНО
1116130.01257-01 12 01-ЛЗ

КОНТЕКСТНЕ ДОСЛІДЖЕННЯ WEB-САЙТУ

Текст програми

1116130.01257-01 12 01

Листів 25

2022

АНОТАЦІЯ

Документ 1116130.01257-01 12 01 «Контекстне дослідження Web-сайту». Текст програми» входить до складу програмної документації до додатку, який надає можливість аналізувати веб-сайт за допомогою методу контекстних досліджень.

В документі містить текст програми. Програма реалізована на мові JavaScript, з використанням каскадних таблиць стилів CSS та мови гіпертекстових документів HTML у програмному середовищі PHPStorm 2019.

ЗМІСТ

1 СХЕМА ВЗАЄМОДІЇ МОДУЛІВ

На рис. 1.1 приведена схема взаємодії програмних модулів.

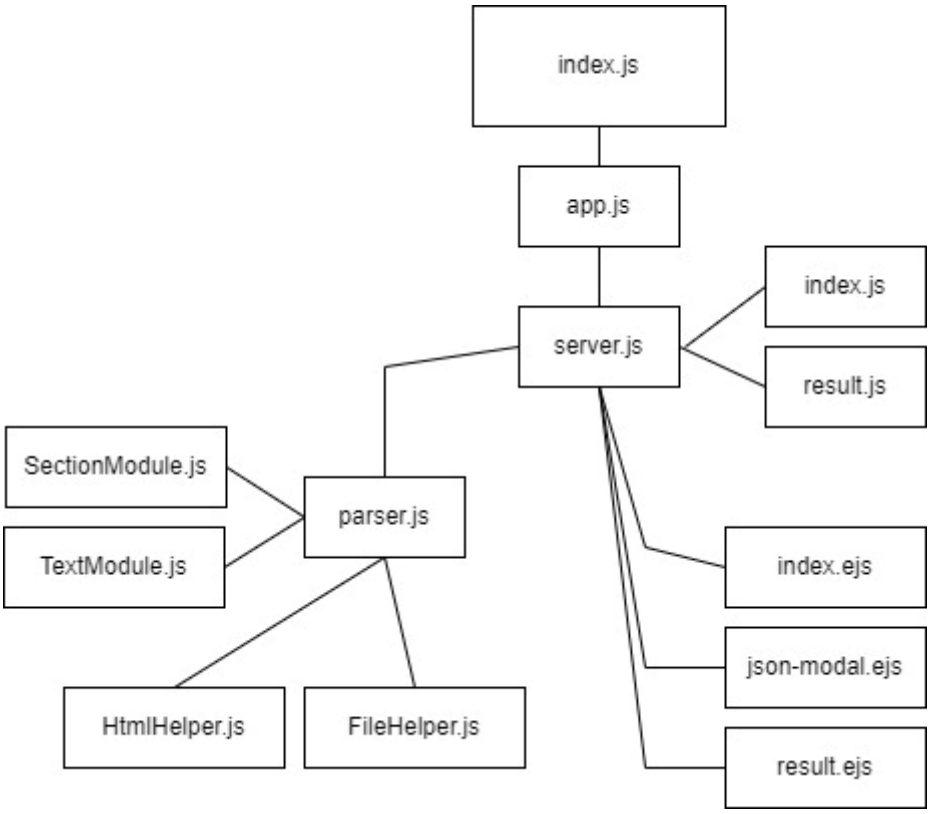


Рисунок 1.1 – Схема взаємодії модулів

2 ТЕКСТ ПРОГРАМИ

2.1 Модуль 1 «index.js»

```
const App = require('./src/modules/app');
const app = new App();
app.run()
```

2.2 Модуль 2 «app.js»

```
const Server = require('./server');
class App {
    constructor() {
        this.server = new Server();
    }
    run() {
        this.server.setEngine();
        this.server.setMiddleWare();
        this.server.initStatic();
        this.server.setRouters();
        this.server.startServer();
    }
}
module.exports = App;
```

2.3 Модуль 3 «server.js»

```
const express = require('express');
const fileUpload = require('express-fileupload');
const Parser = require('./parser');
const { rootDir, createViewPath, joinPath } = require('../helpers/FileHelper');

class Server {
    constructor() {
        this.PORT = 3000;
        this.app = express();
    }

    setEngine() {
        this.app.set('view engine', 'ejs');
    }

    setMiddleWare() {
        // enable file upload
        this.app.use(fileUpload({createParentPath: true}));
    }

    setJsStatic(path) {
        this.app.use('/js', express.static(joinPath(rootDir, path)));
    };

    setCssStatic(path) {
        this.app.use('/css', express.static(joinPath(rootDir, path)));
    }

    initStatic() {
        this.setJsStatic('node_modules/jquery/dist');

        this.setJsStatic('node_modules/bootstrap/dist/js');
        this.setJsStatic('node_modules/jquery.json-viewer/json-viewer');

        this.setJsStatic('node_modules/chart.js/dist');
        this.setJsStatic('node_modules/chartjs-plugin-datalabels/dist');
        this.setJsStatic('/src/web/js');

        this.setCssStatic('node_modules/bootstrap/dist/css');

        this.setCssStatic('node_modules/jquery.json-viewer/json-viewer');
        this.setCssStatic('/src/web/css');
    }

    startServer() {
        let self = this;

        self.app.listen(self.PORT, (error) => {
            error ? console.log(error) : console.log(`listening port ${self.PORT}`);
        });
    }

    setRouters() {
        this.app.get('/', (req, res) => {
            res.render(createViewPath('index'));
        });

        this.app.post('/upload', (req, res) => {
            try {
                let file = req.files.document;
                let mimeType = file.mimetype;
                if ( !(mimeType.indexOf('html') > -1) )
                    res.send('File must be a .html extension!');

                let data = file.data.toString();
                let parser = new Parser(data);

                //res.send(parser.data);
                res.render(createViewPath('result'), {data: JSON.stringify(parser.data)});
            } catch (e) {
                console.log(e);
                res.sendStatus(500);
            }
        });
```

```
    }
}
```

```
module.exports = Server;
```

2.3.1 Модуль 4 «index.js»

```
function initAnimations() {
    const form_block = $('.form-wrapper');
    const text_block = $('.text-wrapper');
    const form_text = $('.form-text');

    $(text_block).hide().fadeIn(1500);

$(form_block).hide().delay(2000).fadeIn(1500);
    $(form_text).hide().delay(3000).fadeIn(1500);
}

function initOnChange() {
    $('#apply').change(() => {
        $('.app-text').toggle();
        $('.process-text').toggle();
        setTimeout(() => $('form').submit(), 2000);
    });
}

$(document).ready(() => {
    initAnimations();
    initOnChange();
});
```

2.3.2 Модуль 5 «result.js»

```
const SELECTOR_WORDS_COUNT = 'wordsCountChart';
const SELECTOR_CORRECT_TEXT = 'correctTextChart';

const SELECTOR_SEMANTIC_TEXT = 'semanticChart';

const COLORS_MAP = [
  '#244D6B', '#78d28a', '#69cfac',
  '#44C1C1','#A557C7', '#8B5EC9',
  '#C144BA','#B65BC8', '#6269CB',
  '#1C5154','#1F485C','#215263'
];

const COLORS_SEMANTIC_MAP = [
  '#349D98','#36A172','#38A86C',
  '#73A437','#3BB03F','#38A85E',
  '#39AC75','#38A89D','#389BA8',
  '#38A8A3','#389DA8','#349D7D'
];

const defaultLegendClickHandler = Chart.defaults.plugins.legend.onClick;
const pieDoughnutLegendClickHandler = Chart.controllers.doughnut.overrides.plugins.legend.onClick;

const newOnClickTextModuleHandler = function (e, legendItem, legend) {
  let canvas = legend.chart.canvas.id;

  let chart = null;

  switch (canvas) {
    case SELECTOR_WORDS_COUNT:
      chart = Chart.getChart(SELECTOR_CORRECT_TEXT);
      break;
    case SELECTOR_CORRECT_TEXT:
      chart = Chart.getChart(SELECTOR_WORDS_COUNT);
      break;
  }

  pieDoughnutLegendClickHandler(e, legendItem, legend);

  if (!chart)
    return false;

  $.each(chart.legend.legendItems, function (index, item) {
    if (item.text === legendItem.text && item.hidden === legendItem.hidden) {
      chart.toggleDataVisibility(index);
      chart.update();
      return true;
    }
  });
};

const newOnClickSemanticModuleHandler = function (e, legendItem, legend) {
  let canvas = legend.chart.canvas.id;

  let chart = null;

  switch (canvas) {
    case SELECTOR_WORDS_COUNT:
      chart = Chart.getChart(SELECTOR_CORRECT_TEXT);
      break;
    case SELECTOR_CORRECT_TEXT:
      chart = Chart.getChart(SELECTOR_WORDS_COUNT);
      break;
  }
```

```
    pieDoughnutLegendClickHandler(e,
legendItem, legend);

    if (!chart)
        return false;

    $.each(chart.legend.legendItems,        function
(index, item) {
        if (item.text === legendItem.text &&
item.hidden === legendItem.hidden) {
            chart.toggleDataVisibility(index);
            chart.update();
            return true;
        }
    });
};

function initCharts(modules) {
    setTextModuleChart(modules.TextModule);
    setSectionsModule(modules.SectionsModule);
}

function                getRandomRGBArray(count,
colors_map) {
    let colors = [];

    for (let i = 0; i < count; i++)
        colors.push(colors_map[i]);

    return colors;
}

function getRandomRGB() {
    const        randomNum        =        ()        =>
Math.floor(Math.random() * (235 - 52 + 1) + 52);

    const        randomRGB        =        ()        =>
`rgb(${randomNum()},        ${randomNum()},
${randomNum()})`;

    return randomRGB();
}

function setChart(params) {
    let type            = params.type;
    let paramsData      = params.data;
    let         backgroundColor                    =
params.backgroundColors;
    let selector        = params.selector;
    let onClickCallback  = params.callback;
    let datalabels      = params.datalabels;

    let chart = $(selector);

    if (!backgroundColor)
        backgroundColor                            =
getRandomRGBArray(paramsData.data.length,
COLORS_MAP);

    const chartData = {
        plugins: [ChartDataLabels],
        labels: paramsData.labels,
        datasets: [{
            data: paramsData.data,
            backgroundColor: backgroundColor,
            hoverOffset: 4
        }]
    };

    let legend = {};
    if (onClickCallback)
        legend.onClick = onClickCallback;

    new Chart(chart, {
        plugins: [ChartDataLabels],
        type: type,
        data: chartData,
        options: {
            plugins: {
              legend: legend,
              tooltip: {
                  enabled: false,
              },
              datalabels: datalabels
            },
            responsive: false,
        }
    });
}
```

```
function setJson(selector, data) {
   $(selector).attr('data-json',
JSON.stringify(data));
}

function setSectionsModule(module) {
   const backgroundColors =
getRandomRGBArray(module.charts.semanticT
ags.data.length, COLORS_SEMANTIC_MAP);

   const paramsSemantic = {
      type: 'doughnut',
      selector:
`#${SELECTOR_SEMANTIC_TEXT}`,
      data: module.charts.semanticTags,
      backgroundColors: backgroundColors,
      callback: null,
      datalabels: {
         color: '#faffec',
      }
   };

   setChart(paramsSemantic);
   setJson('.semantic-data',
module.data.semantic);
}

function setTextModuleChart(module) {
   const backgroundColors =
getRandomRGBArray(module.charts.correctTex
t.data.length, COLORS_MAP);

   const paramsCorrect = {
      type: 'doughnut',
      selector:
`#${SELECTOR_CORRECT_TEXT}`,
      data: module.charts.correctText,
      backgroundColors: backgroundColors,
      callback: newOnClickTextModuleHandler,
      datalabels: {
         color: '#faffec',
         formatter: function(value) {
            return `${value} %`;
         }
      }
   };

   const paramsWords = {
      type: 'doughnut',
      selector:
`#${SELECTOR_WORDS_COUNT}`,
      data: module.charts.countWordsTotal,
      backgroundColors: backgroundColors,
      callback: newOnClickTextModuleHandler,
      datalabels: {
         color: '#faffec',
      }
   };

   setChart(paramsCorrect);
   setChart(paramsWords);
   setJson('.readable-data',
module.data.readable);
}

function initGlobalCharts() {
   Chart.defaults.plugins.legend.labels.color =
'white';
}

function initJsonViewer() {
   $('.show-json').on('click', function (e) {
      e.preventDefault();

      let json_data = JSON.parse($(this).attr('data-
json'));
      $('#json-renderer').jsonViewer(json_data,
{collapsed: true});
      $('#json-modal').modal('show');
   });
}

$(document).ready(() => {
   const data =
JSON.parse($('input[name="data"]').val());
   console.log(data);
   initGlobalCharts();
   initJsonViewer();
   initCharts(data);
}
```

2.3.3 Модуль 6 «index.ejs»

```
<!doctype html>
<html lang="ru">
<head>

  <script src="js/jquery.min.js"></script>
  <script src="js/index.js"></script>

  <link rel="stylesheet" href="css/bootstrap.min.css">
  <link rel="stylesheet" href="css/index.css">

  <meta charset="UTF-8">
  <meta name="viewport"
      content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Web-App | Main</title>
</head>
<body>
<div class="body-index">
  <div class="content-wrapper">
    <div class="text-wrapper">
      <h2 class="app-text">Перший веб-додаток з використанням методу контекстного дослідження</h2>
      <h2 class="process-text">Запущен процесс...</h2>
    </div>
    <div class="form-wrapper">
      <form id="upload-form" action="/upload" method="post" enctype="multipart/form-data">
        <label for="apply"><input type="file" name="document" id="apply" accept=".html">Завантажити документ</label>
        <p class="form-text">Тільки *.html підтримуються</p>
      </form>
    </div>
  </div>
</div>
</body>
</html>
```

2.3.4 Модуль 7 «json-modal.ejs»

```
<!-- Modal -->
<div class="modal fade" id="json-modal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog modal-lg" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLabel">JSON Viewer</h5>
      </div>
      <div class="modal-body">
        <div id="json-renderer"></div>
      </div>
    </div>
  </div>
</div>
```

2.3.5 Модуль 8 «result.ejs»

```
<!doctype html>
<html lang="ru">
<head>

  <script src="js/jquery.min.js"></script>
  <script src="js/chart.min.js"></script>
  <script src="js/chartjs-plugin-datalabels.js"></script>
  <script src="js/result.js"></script>
  <script src="js/jquery.json-viewer.js"></script>
  <script src="js/bootstrap.js"></script>

  <link rel="stylesheet" href="css/bootstrap.min.css">
  <link rel="stylesheet" href="css/jquery.json-viewer.css">
  <link rel="stylesheet" href="css/index.css">

  <meta charset="UTF-8">
  <meta name="viewport"
      content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">

  <title>Web-App | Results</title>
</head>
<body>
<div class="body-result">
  <div>
    <input name="data" type="hidden" value="<%= data %>">
  </div>
  <div class="content-wrapper">

    <%- include('./json-modal'); %>

    <div class="module-block-item text-module">
      <div class="module-block readable-module">
        <h2>Читабельність тексту</h2>
        <p class="form-text text-center">Визначається за різними показниками</p>
        <p class="form-text text-center">(чим вищий відсоток читабельності, тим краще)</p>
        <canvas id="correctTextChart"></canvas>
        <p class="module-text">*враховуються тільки теги, що є на сторінці і мають відсоток читабельності більше 0</p>
        <a href="#" class="readable-data show-json text-center">Продивитись дані</a>
      </div>
      <div class="module-block words-module">
        <h2>Кількість слів в тегах</h2>
        <p class="form-text text-center"></p>
        <canvas id="wordsCountChart"></canvas>
        <p class="module-text">*враховуються тільки теги, що є на сторінці і мають контент всередині себе</p>
        <a href="#" class="readable-data show-json text-center">Продивитись дані</a>
      </div>
    </div>
    <div class="module-block-item semantic-module">
      <div class="module-block tagCountBlock">
        <h2>Семантичний аналіз</h2>
        <p class="form-text text-center">Наявність семантичних тегів</p>
        <p class="form-text text-center">(семантичні теги покращують позиції в видачі пошуковими системами)</p>
        <canvas id="semanticChart"></canvas>
```

```
            <p class="module-text">*враховуються тільки теги, що є на сторінці</p>
            <a href="#" class="semantic-data show-json text-center">Продивитись дані</a>
          </div>
        </div>

        <footer>
```

```
            <div class="footer-wrapper">

            </div>
          </footer>
        </div>
      </div>
      </body>
      </html>
```

2.4 Модуль 9 «parser.js»

```
const { htmlTojQueryDoc } = require('../helpers/HtmlHelper');
const TextModule = require('./parser-modules/TextModule');
const SectionsModule = require('./parser-modules/SectionsModule');

const MODULES_MAP = {
   TextModule,
   SectionsModule
};

class Parser
{
   constructor(html) {
      this.document = htmlTojQueryDoc(html);
      this.data = {};
      this.run();
   }

   run() {
      let self = this;

      for (let moduleName in MODULES_MAP) {
         let module = new MODULES_MAP[moduleName](self.document);
         this.data[moduleName] = module.run();
      }
   }
}

module.exports = Parser;
```

2.4.1 Модуль 10 «SectionModule.js»

```
const { JSDOM } = require( "jsdom" );
const { window } = new JSDOM( "" );
const $ = require( "jquery" )( window );

const SEMANTIC_TAGS = [
   'article',
   'aside',
   'details',
   'figcaption',
   'figure',
   'footer',
   'header',
   'main',
   'mark',
   'nav',
   'section',
   'summary',
   'time',
];

const BASIC_SEMANTIC_TAGS = [
   'nav', 'main', 'footer'
];

class SectionsModule {

   constructor(document) {
      this.document = document;
      this.tags = this.getTagsMap();
   }

   getTagsMap() {
      return [].concat.apply([], [
         SEMANTIC_TAGS,
      ]);
   }

   getSemanticData() {
      let self = this;

      let tags = {};

      let items = [];

      let total_tags = 0;
      let total_unique_tags = 0;
      let tags_basic_count = 0;
      $.each(self.tags, function (tag_i,
tag_selector) {
         let tag_items =
self.document.find(tag_selector);
         let tag_items_count = tag_items.length;
         if (!tag_items_count)
            return;

//console.log(`${tag_selector}|${tag_items_coun
t}`); // debug

         items = [];
         $.each(tag_items, function (item_i,
tag_item) {
            let tag_text = $(tag_item).text();
            items.push({
               text: tag_text,
               text_length: tag_text.length,
               text_words_count:
tag_text.split('
').length,
            });
         });

         let isBasicTag =
BASIC_SEMANTIC_TAGS.includes(tag_select
or);
         if (isBasicTag)
            tags_basic_count++;

         tags[tag_selector] = {
            items: items,
            count: items.length,
            is_basic: isBasicTag
         };

         total_unique_tags++;
         total_tags+=tag_items_count;
      });

      return {
```

```
        tags: tags,
        total_tags: total_tags,
        total_unique_tags: total_unique_tags,
        tags_basic_total:
BASIC_SEMANTIC_TAGS.length,
        tags_basic_count:
BASIC_SEMANTIC_TAGS.length,
      };
    }

    getData() {
      let data = {};

      data.semantic = this.getSemanticData();

      return data;
    }

    getSemanticTagsChartData(data) {
      let tags = data.semantic.tags;
      let tag_name_map = [];
      let tag_count_map = [];

      $.each(tags, function (tag_name, tag_item) {
        if (tag_item.count === 0 ||
tag_item.total_readable === 0)
          return;

        let tag_name_count = `<${tag_name}>`;
        tag_name_map.push(tag_name_count);
        tag_count_map.push(tag_item.count);
      });

      return {
        labels: tag_name_map,
        data: tag_count_map
      }
    }

    run() {
      let data = this.getData();

      let correctText =
this.getSemanticTagsChartData(data);

      let charts = {
        semanticTags: correctText,
      };

      return {
        data: data,
        charts: charts,
      }
    }
}

module.exports = SectionsModule;
```

2.4.2 Модуль 11 «TextModule.js»

```
const { JSDOM } = require( "jsdom" );
const { window } = new JSDOM( "" );
const $ = require( "jquery" )( window );

const LONG_TAGS_LIST = [
   'p', 'article',
];

const SHORT_TAGS_LIST = [
   'span', 'b', 'strong', 'abbr', 'address',
];

const TITLE_TAGS_LIST = [
   'h1', 'h2', 'h3', 'h4',
];

const CAPTION_TAGS_LIST = [
  'caption', 'label', 'legend'
];

const QUOTE_TAGS_LIST = [
   'blockquote',
];

const LINE_TAGS_LIST = [
   'li', 'ol'
];

const LONG_TAGS_WORDS_MIN    = 100;
const LONG_TAGS_WORDS_MAX    = 200;

const SHORT_TAGS_WORDS_MIN   = 25;
const SHORT_TAGS_WORDS_MAX   = 50;

const TITLE_LENGTH_MIN       = 60;
const TITLE_LENGTH_MAX       = 70;
// OR
const TITLE_WORDS_MIN        = 10;
const TITLE_WORDS_MAX        = 25;

const CAPTION_LENGTH_MAX     = 50;
const QUOTE_WORDS_MIN        = 40;
const LINE_LENGTH_MAX        = 70;
```

```
class TextModule {
   constructor(document) {
      this.document = document;
      this.tags = this.getTagsMap();
   }

   getTagsMap() {
     return [].concat.apply([], [
        LONG_TAGS_LIST,
        SHORT_TAGS_LIST,
        TITLE_TAGS_LIST,
        CAPTION_TAGS_LIST,
        QUOTE_TAGS_LIST,
        LINE_TAGS_LIST,
      ]);
   }

   getData() {
      let data = {};

      data.readable = this.getReadableData();

      return data;
   }

   getTextCountWords(text) {
      return text ? text.split(' ').length : 0;
   }

   getTextLength(text) {
      return text ? text.length : 0;
   }

   formatText(text) {
      // remove escape line
      let formatted_text = text.replace(/[\\n ]/g,
'').trim();
      // remove double spaces
      formatted_text =
formatted_text.replace(/\s{2,}/g,' ');
      // return formatted
      return formatted_text;
```

```
  }

  isTextReadable(text, selector) {
    let self = this;

    let words_count = self.getTextCountWords(text);
    let text_length = self.getTextLength(text);

    // long
    if (LONG_TAGS_LIST.includes(selector))
      return words_count >= LONG_TAGS_WORDS_MIN && words_count <= LONG_TAGS_WORDS_MAX;

    if (SHORT_TAGS_LIST.includes(selector))
      return words_count >= SHORT_TAGS_WORDS_MIN && words_count <= SHORT_TAGS_WORDS_MAX;

    if (TITLE_TAGS_LIST.includes(selector))
      return (words_count >= TITLE_WORDS_MIN && words_count <= TITLE_WORDS_MAX) || (text_length >= TITLE_LENGTH_MIN && text_length <= TITLE_LENGTH_MAX);

    if (CAPTION_TAGS_LIST.includes(selector))
      return text_length <= CAPTION_LENGTH_MAX;

    if (QUOTE_TAGS_LIST.includes(selector))
      return words_count >= QUOTE_WORDS_MIN;

    if (LINE_TAGS_LIST.includes(selector))
      return text_length <= LINE_LENGTH_MAX;

    return false;
  }

  getReadableData() {
    let self = this;

    let total_words_count = 0;

    let tag_data_arr = {};

    $.each(self.tags, (tag_selector_i, tag_selector) => {
      let all_tags = $(self.document).find(tag_selector).filter(':not(:empty)');

      let items = [];
      let tag_text_total = [];
      let tag_words_count = 0;

      let tag_is_normal = 0;

      $.each(all_tags, (tag_i, tag_item) => {
        let tag_text = self.formatText($(tag_item).text());
        if (!tag_text)
          return;

        let text_length = self.getTextLength(tag_text);
        let text_words_count = self.getTextCountWords(tag_text);

        let is_readable = self.isTextReadable(tag_text, tag_selector);

        if (is_readable)
          tag_is_normal++;

        tag_words_count += text_words_count;
        total_words_count += text_words_count;
        tag_text_total.push(tag_text);

        items.push({
          text: tag_text,
```

```
                text_length: text_length,
                text_words_count:
text_words_count,
                is_readable: is_readable,
            });
        });

        tag_data_arr[tag_selector] = {
            items: items,
            count: items.length,
            total_text: tag_text_total.join(' '),
            total_words_count: tag_words_count,
            total_readable:      tag_is_normal      ?
(tag_is_normal / items.length * 100).toFixed(2) :
0,
        };
    });

    return {
        tags: tag_data_arr,
        total_words: total_words_count
    };
  }

  getCorrectTagsChartData(data) {
    let tags = data.readable.tags;
    let tag_name_map = [];
    let tag_count_map = [];

    $.each(tags, function (tag_name, tag_item) {
        if      (tag_item.count      ===      0      ||
tag_item.total_readable === 0)
            return;

        //let tag_name_count = `<${tag_name}>
( ${tag_item.count} )`;
        let tag_name_count = `<${tag_name}>`;
        tag_name_map.push(tag_name_count);

tag_count_map.push(tag_item.total_readable);
    });

    return {
        labels: tag_name_map,
        data: tag_count_map
    }
  }

  getCountWordsChartData(data) {
    let tags = data.readable.tags;
    let tag_name_map = [];
    let tag_count_map = [];

    $.each(tags, function (tag_name, tag_item) {
        if      (tag_item.count      ==      0      ||
tag_item.total_readable      ===      0      ||
tag_item.total_words_count == 0)
            return;

        //let tag_name_count = `<${tag_name}>
( ${tag_item.count} )`;
        let tag_name_count = `<${tag_name}>`;
        tag_name_map.push(tag_name_count);

tag_count_map.push(tag_item.total_words_coun
t);
    });

    return {
        labels: tag_name_map,
        data: tag_count_map
    }
  }

  run() {

    let data = this.getData();
    let              correctText              =
this.getCorrectTagsChartData(data);
    let              countWordsChart              =
this.getCountWordsChartData(data);

    let charts = {
      correctText: correctText,
      countWordsTotal: countWordsChart,
    };

    return {
        data: data,
        charts: charts
```

```
        };
    }
}
```

```
module.exports = TextModule;
```

2.4.3 Модуль 12 «HtmlHelper.js»

```
const { JSDOM } = require( "jsdom" );
const { window } = new JSDOM( "" );
const $ = require( "jquery" )( window );

const htmlTojQueryDoc = (html) => $(html);

module.exports = {
    htmlTojQueryDoc
};
```

2.4.4 Модуль 13 «FileHelper.js»

```
const path = require('path');

const rootDir = path.resolve(__dirname, '../../');

const createViewPath = (page) =>
path.resolve(__dirname, '../ejs-views',
`${page}.ejs`);

const joinPath = (first, second) => path.join(first,
second);

module.exports = {
   rootDir,
   createViewPath,
   joinPath,
};
```

ЗАТВЕРДЖЕНО
1116130.01257-01 13 01-ЛЗ

КОНТЕКСТНЕ ДОСЛІДЖЕННЯ WEB-САЙТУ

Опис програми

1116130.01257-01 13 01

Аркушів 21

2022

АНОТАЦІЯ

Документ 1116130. 01257-01 13 01 «Контекстне дослідження Web-сайту». Опис програми» входить до складу програмної документації до додатку, який надає можливість проводити аналіз веб-сайту за допомогою методу контекстних досліджень.

У даному документі представлений опис програми системи: функціональне призначення, опис логічної структури, використані технічні засоби, виклик і завантаження, вхідні і вихідні дані, опис інтерфейсу користувача, порядок роботи з програмою. Програма реалізована на мові JavaScript, з використанням каскадних таблиць стилів CSS та мови гіпертекстових документів HTML у програмному середовищі PHPStorm 2019. Для роботи з додатком необхідний смартфон або комп’ютер з встановленим Chrome 60 і вище, доступ до мережі Інтернет.

ЗМІСТ

1 ЗАГАЛЬНІ ВІДОМОСТІ

Додаток «Контекстне дослідження Web-сайту» призначений для аналізу веб-сайтів за допомогою методів контекстних досліджень.

Для функціонування даного програмного продукту необхідно: Chrome 60, актуальна бібліотека NodeJs.

В документі містить текст програми. Програма реалізована на мові JavaScript, з використанням каскадних таблиць стилів CSS та мови гіпертекстових документів HTML у програмному середовищі PHPStorm 2019.

2 ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Функціональне призначення розробки:

– аналіз гіпертекстових документів за допомогою методу контекстних досліджень;

– можливість аналізу за допомогою визначених критеріїв.

3 ОПИС ЛОГІЧНОЇ СТРУКТУРИ

3.1 Алгоритм програми

На рис. 3.1 представлено алгоритм роботи програми.

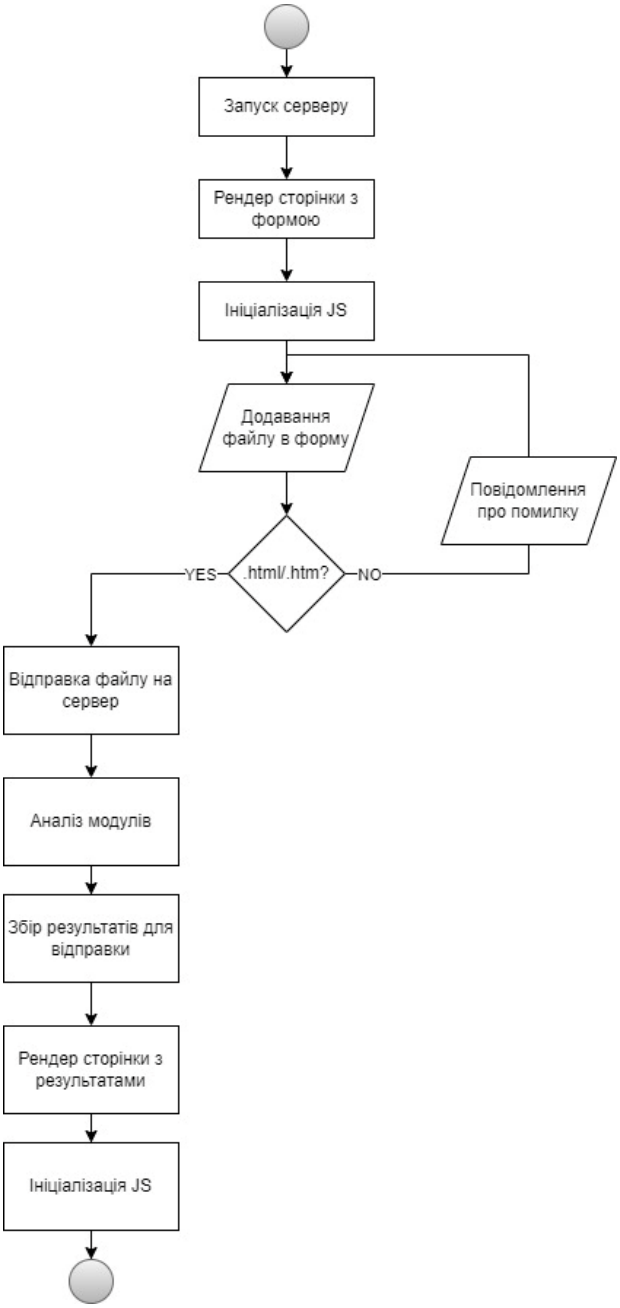


Рисунок 3.1 – Алгоритм роботи програми

3.2. Використані методи

Програма використовує наступні методи:

– основна робота програми написана засобами мови JavaScript, що відповідає за клієнтську та серверну частини;

– для сторінок використовується формат файлу .ejs, з використанням розмітки гіпертекстових документів HTML та мови програмування JavaScript;

– для роботи сервера використовується Express.

3.3 Структура програми

На рис. 3.2 відображені діаграма класів основних програмних модулів та взаємозв‘язок між ними.

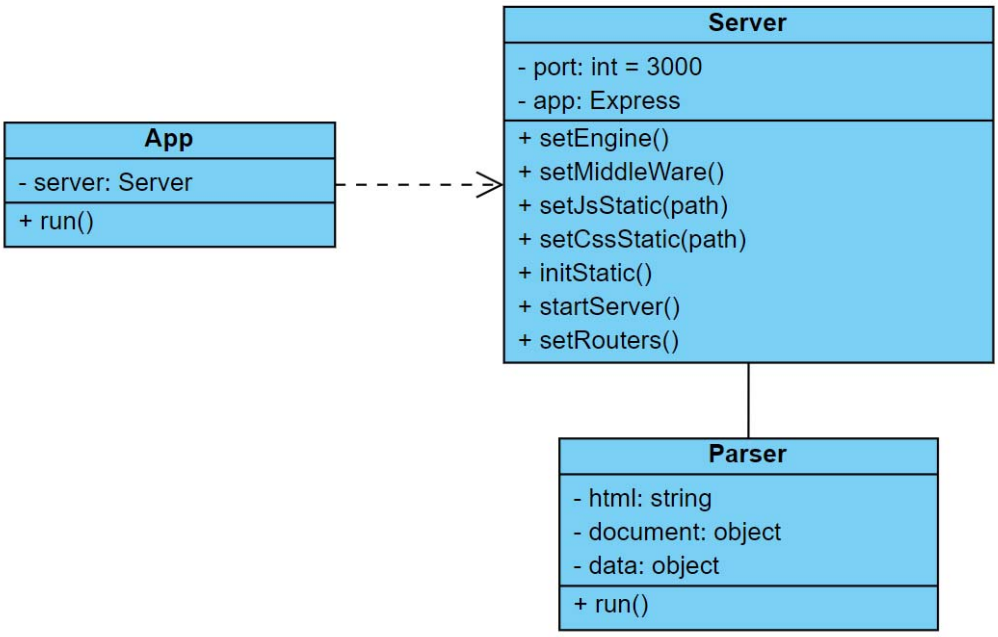


Рисунок 3.2 – Діаграма класів основних програмних модулів

Програмні модулі виконують наступні функції:

– app.js – виконує ініціалізацію модулів для роботи з додатком;

– server.js – виконує запуск серверу, встановляє правила, маршрути для запитів, визначає файли стилів, скриптів;

– parser.js – виконує запуск модулів для обробки даних гіпертекстового документу.

3.4 Зв‘язки програми з іншими програмами

Додаток складається з двох частин, а саме: клієнтської частини та серверу. Клієнтська частина відповідає за взаємодію користувача з клієнтським додатком.

Сервер відповідає за зберігання та обробку інформації, що надходить з клієнтської частини. Взаємодія цих частин відбувається за допомогою протоколу HTTP.

4 ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Мінімальна конфігурація комп'ютера для забезпечення працездатності програмного продукту:

- Intel Pentium IV з частотою процесора від 2.0 Ггц;
- ОЗУ 256 Мб;
- 20 Гб на жорсткому диску;
- дисковод 3.5", 1.4 Мбайт;
- CD-ROM;
- дисплей VGA;
- принтер струменевий / лазерний ;
- стандартна клавіатура (104/105 клавіші);
- маніпулятор "миша".

5 ВИКЛИК ТА ЗАВАНТАЖЕННЯ

Для початку роботи з додатком «Контекстне дослідження Web-сайту», необхідно запустити додаток командою npm run nodemon-dev, див. рис. 5.1.

Рисунок 5.1 – Термінал виклику додатку

Після очікування розгортання додатку, необхідно перейти в браузер та в поле адреси ввести localhost:3000, після чого з’явиться стартовий екран, див. рис. 5.2.

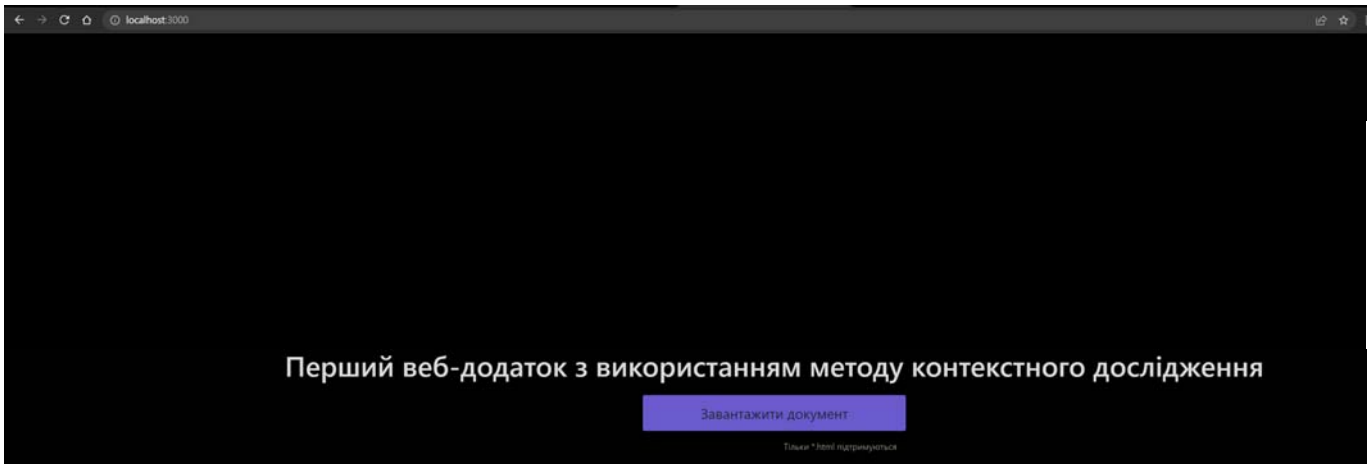


Рисунок 5.2 – Вікно браузера із стартовим екраном

6 ВХІДНІ ДАНІ

Вхідними даними додатку є:

– гіпертекстовий документ в форматі .html, .htm.

7 ВИХІДНІ ДАНІ

Вихідними даними додатку є:

– результат аналізу парсеру за модулями, які налаштовані в проекті, з можливістю їх перегляду в інтерактивному форматі та представлені у вигляді тексту, а також у вигляді графіків.

8 ОПИС ІНТЕРФЕЙСУ КОРИСТУВАЧА

8.1 Опис станів програми

Після запуску програми на екрані з’являється стартовий екран, див. рис. 5.2. Сторінка складається з тексту головної сторінки, з зазначенням використовуваного додатку та кнопки для завантаження файлу, а також підпис з поміткою про формат завантажуваного файлу, див. рис. 8.1.

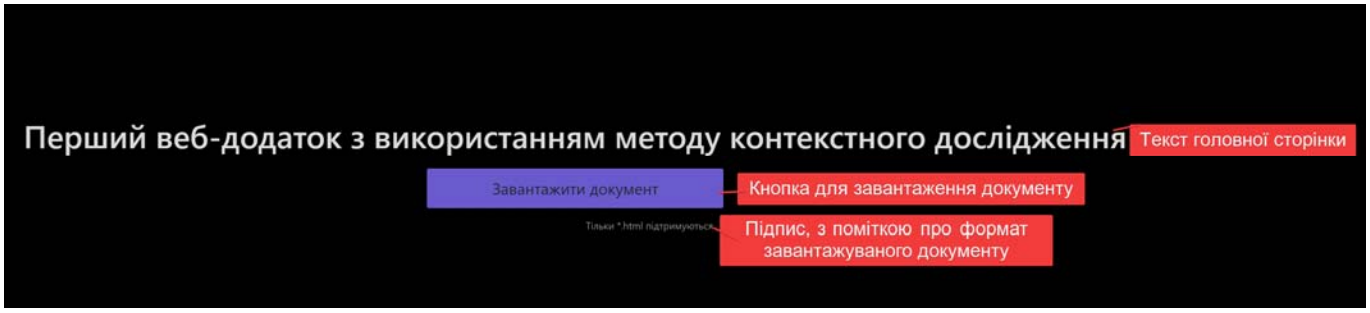


Рисунок 8.1 – Елементи головної сторінки

Для завантаження файлу на сервер, необхідно натиснути кнопку «Завантажити документ» – після натискання, відкриється екранна форма провідника системи, в якому буде налаштований фільтр на файли з розширенням .html та .htm, див. рис. 8.2.

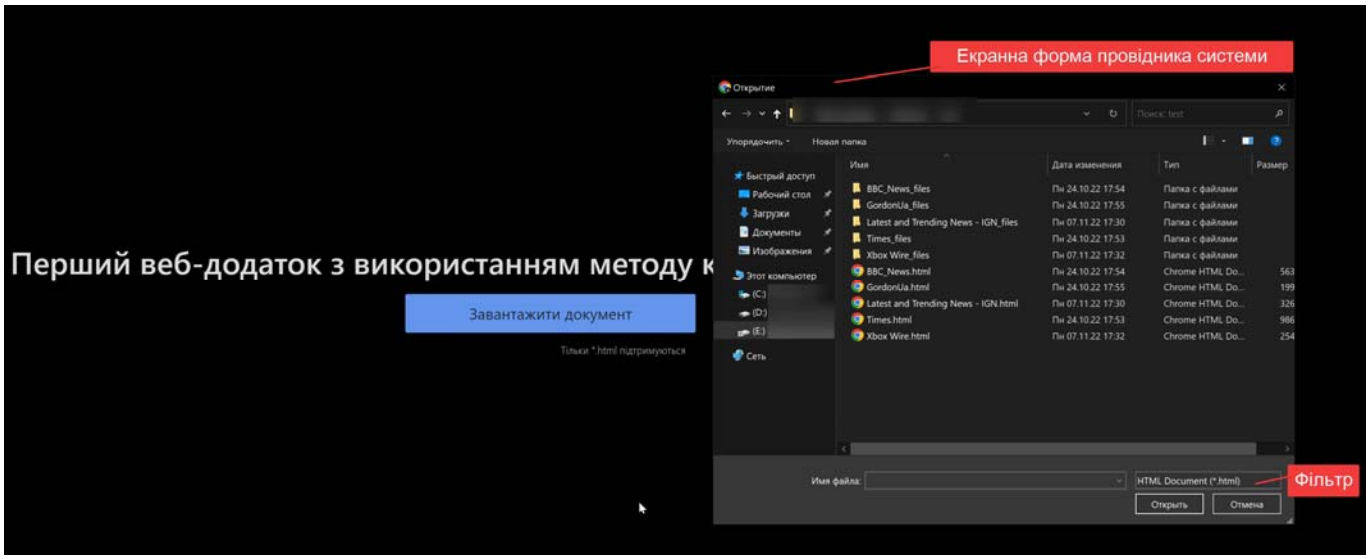


Рисунок 8.2 – Елементи при натисканні на кнопку завантаження документу

Після вибору файлу, необхідно підтвердити вибір. Після цього, екранна форма закривається, текст сторінки змінює свою назву, а кнопка «Завантажити документ» – блокується, для запобігання помилок, див. рис. 8.3.

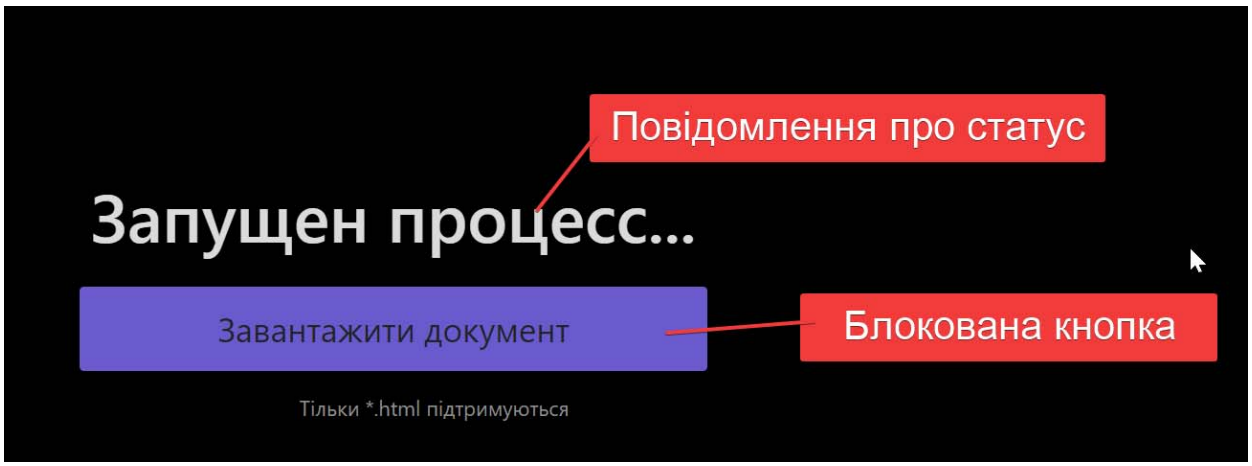


Рисунок 8.3 – Елементи при завантажені файлу на сервер

Після вибору в екранній формі, гіпертекстовий документ завантажується на сервер, на якому виконується аналіз його аналіз. Під час аналізу, сторінка не оновлюється. Після завершення роботи парсеру, виконується переадресація на сторінку з результатами, див. рис. 8.4.

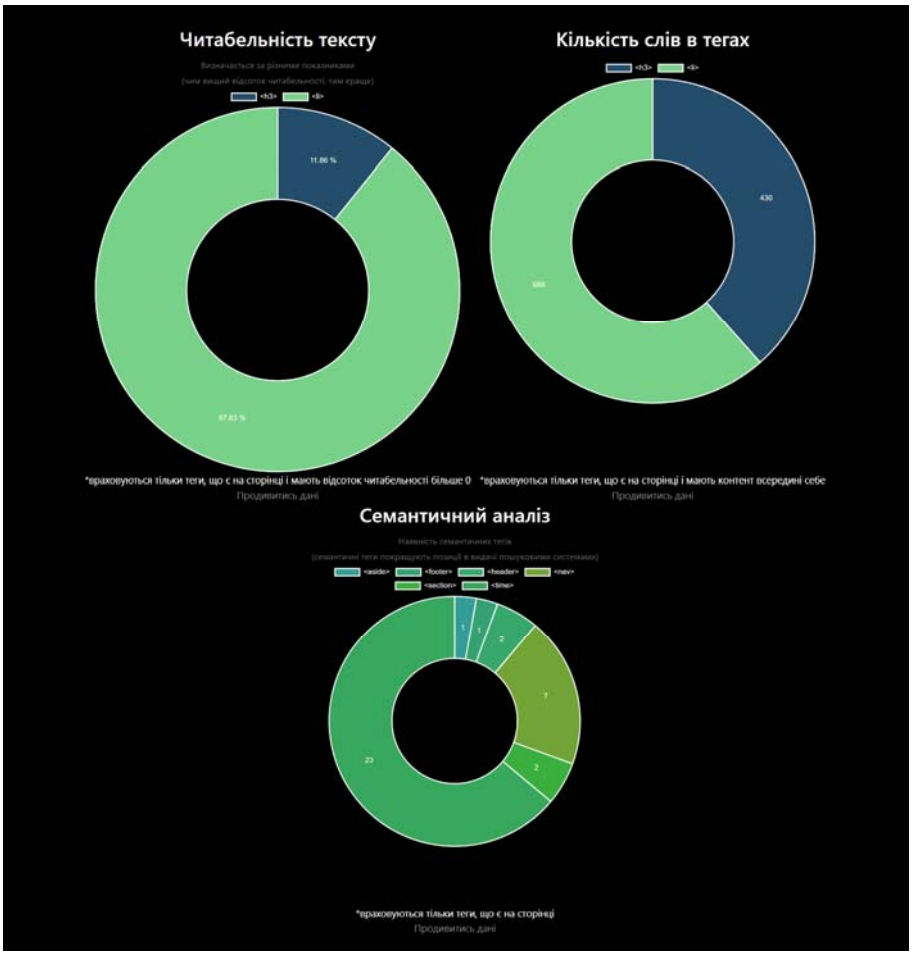


Рисунок 8.4 – Сторінка результатів аналізу

Сторінка результатів складається з результатів аналізованих даних, представлених у вигляді кругових діаграм, з описом модулю, а також можливістю перегляду текстового представлення, див. рис. 8.5.

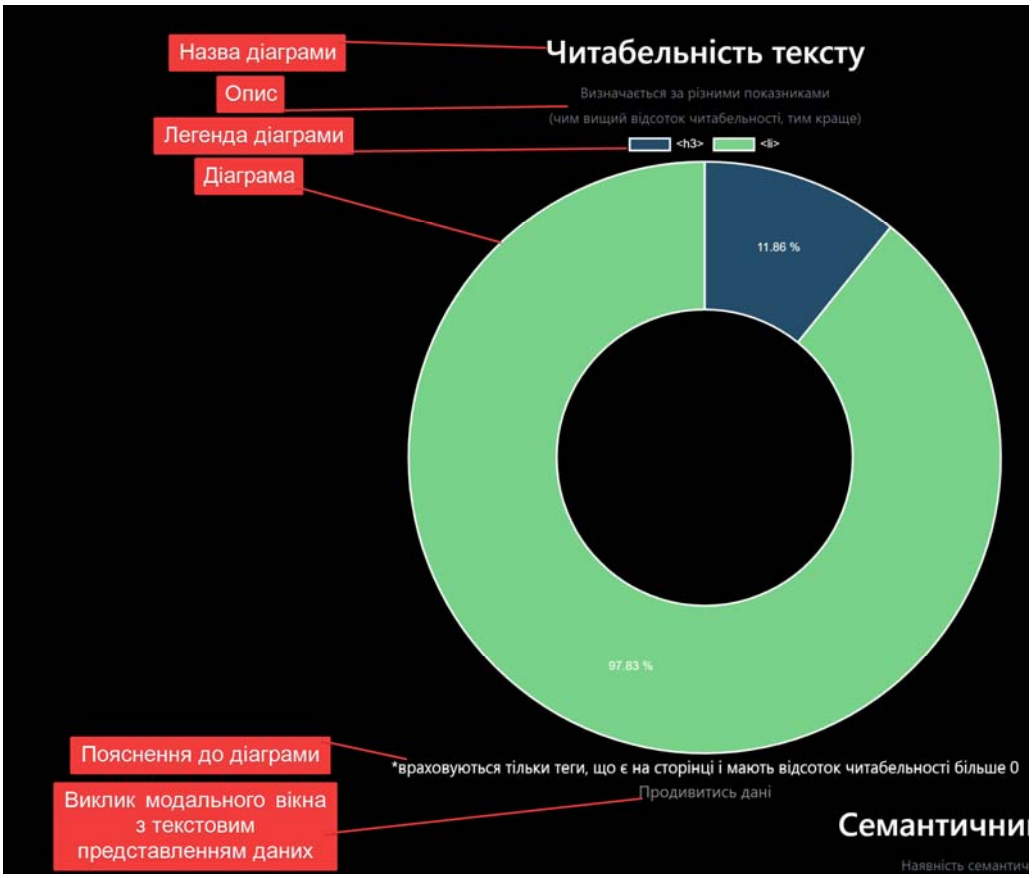


Рисунок 8.5 – Елементи діаграми на сторінці результатів

При натисканні на текст «Продивитися дані» – відкривається модальне вікно, з текстовим представленням результатів, у вигляді дерева у форматі JSON, див. рис. 8.6.

Рисунок 8.6 – Вікно текстового представлення даних

У вікні можливо переглядати результати в форматі JSON, представленого у вигляді дерева, з відкритими та закритими елементами, полями та значеннями, див. рис. 8.7.

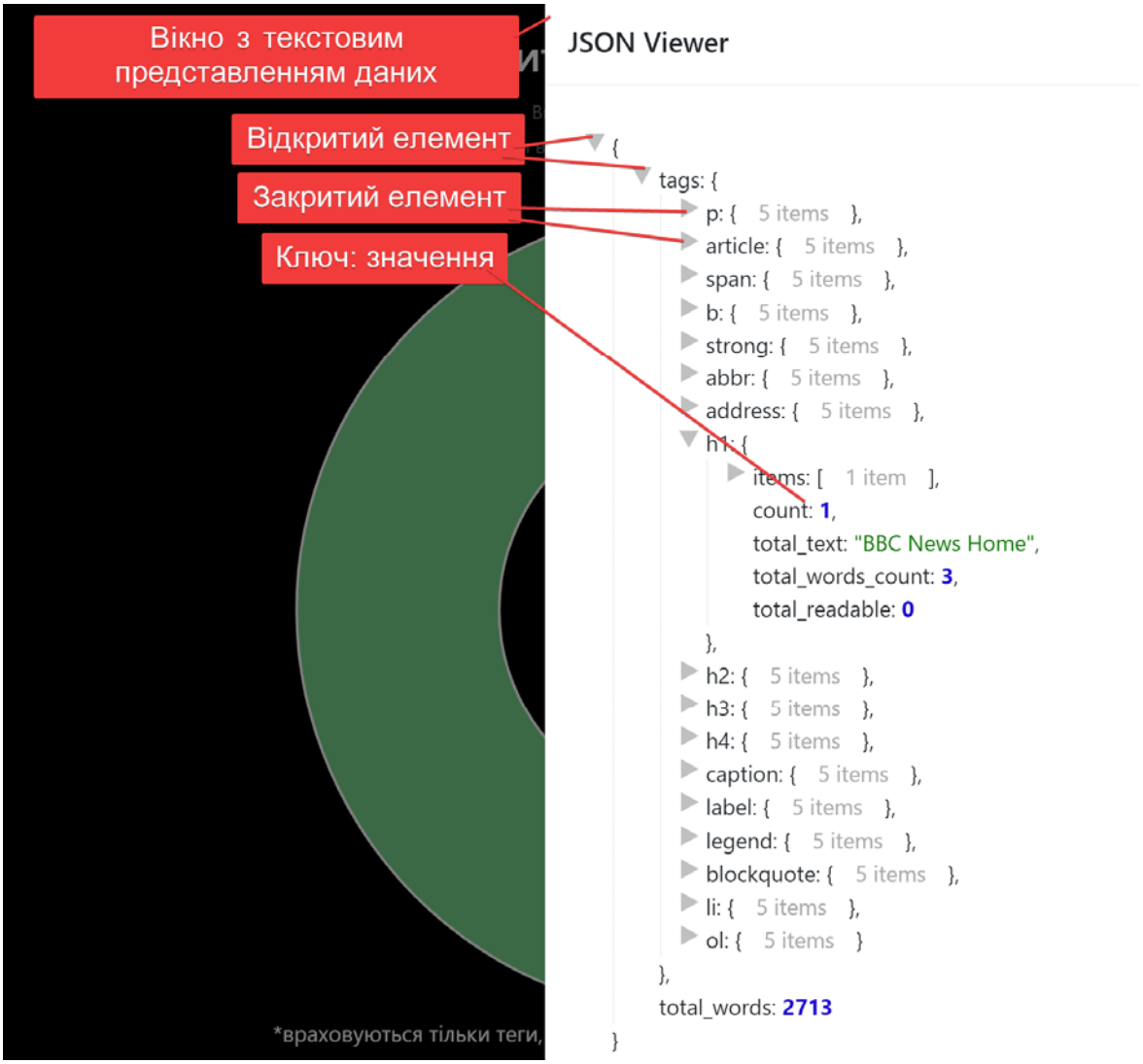


Рисунок 8.7 – Елементи вікна текстового представлення даних

Для закриття модального вікна, необхідно натиснути на область за межами модального вікна.

8.2 Опис керування діалогом

Діалог між користувачем та додатком відбувається за допомогою інтерфейсу користувача, а саме користувач взаємодіє за допомогою кнопок та елементів на сторінці.

8.3 Формування екранів

Формування екранів додатку відбувається під час діалогу між користувачем та інтерфейсом користувача.

Для завершення роботи з програмою користувач повинен закрити вкладку браузера або сам браузер, див. рис. 5.2.

9 ПОРЯДОК РОБОТИ З ПРОГРАМОЮ

Технологія роботи і порядок виконання моделі:

1) Запустити додаток командою npm run nodemon-dev, очікувати повідомлення в терміналі.

2) Відкрити браузер.

3) В адресне поле ввести localhost:3000.

4) На стартовій сторінці, натиснути кнопку «Завантажити документ».

5) Після натискання, в екранній формі провідника системи, обрати гіпертекстовий файл, підтвердити вибір.

6) Очікувати, поки сервер проводить аналіз, після чого, буде проведена переадресація.

7) На сторінці з результатами, переглянути результати аналізу. За необхідністю – переглянути текстове представлення, натиснувши кнопку під бажаною діаграмою «Продивитися дані».

10 ПОВІДОМЛЕННЯ

У табл. 10.1 наведені повідомлення користувачу, що можуть з’явитися під час роботи з програмою.

Таблиця 10.1 – Повідомлення користувачу

Текст повідомлення	Опис ситуації	Рекомендовані дії
Відсутнє підключення до мережі інтернет.	На пристрої відсутній Інтернет.	Ввімкнути на пристрої доступ до мережі Інтернет.
Помилка під час виконання дії	Запит на виконання деякої дії закінчився невдачею	Виконати рекомендовані дії для усунення несправності або звернутися в службу підтримки
Запущен процес…	Проходить аналіз завантажуваного гіпертекстового документу	Блокування кнопки «Завантаження документу»
Формат завантажуваного файлу відмінний від .html/.htm	Користувач неправильно обрав файл, у випадку неправильної роботи фільтру екранної форми провідника системи	Повідомлення про неправильний формат файлу

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Івченко, Ю.М. Основи стандартизації програмних систем [Текст]: методичні вказівки до дипломного проектування та лабораторних робіт / уклад.: Ю. М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. тра-нсп. ім. акад. В. Лазаряна, 2009. - 38 с.

ЗАТВЕРДЖЕНО
1116130.01257-01 ІЗ 01-ЛЗ

КОНТЕКСТНЕ ДОСЛІДЖЕННЯ WEB-САЙТУ

Керівництво користувача

1116130.01257-01 ІЗ 01

Листів 12

2022

АНОТАЦІЯ

Документ 1116130.01206-01 ІЗ 01 «Контекстне дослідження Web-сайту» є керівництвом користувача програмного додатку «Контекстне дослідження Web-сайту», та входить до складу документації на проект.

Контекстне дослідження гіпертекстових сторінок є необхідним для аналізу читабельності сторінок, перегляду семантичних тегів, кількісної та якісної характеристики, при проведені семантичного аналізу.

В документі містяться основні вимоги до розробки програмного додатку та його функціональні можливості. об’єм оперативної пам’яті, що займає програмний комплекс складає до 100 мб. конфігурація комп’ютера стандартна. програма функціонує в середовищі Windows 10.

ЗМІСТ

ВСТУП

Розроблений додаток має назву «Контекстне дослідження Web-сайту».

Додаток був розрахований на функціонування в операційних системах MS Windows 7/8/8.1/10, а також в Unix системах.

Додаток застосовується для аналізу гіпертекстових сторінок з метою аналізу за допомогою методу контекстних досліджень за визначеними критеріями.

Додаток надає такі можливості:

– аналіз гіпертекстових документів за допомогою методу контекстних досліджень;

– можливість аналізу за допомогою визначених критеріїв.

Користувачу не потрібні знання в програмуванні – лише навички роботи з персональним комп’ютером та керівництво користувача.

2 ПРИЗНАЧЕННЯ ТА УМОВИ ЗАСТОСУВАННЯ

Додаток призначений для аналізу гіпертекстових сторінок за допомогою контекстного дослідження для розробників для спеціалістів, що займаються веб-розробкою.

Програмний додаток може працювати в таких операційних системах:

— Microsoft Windows, 10, x86-32 і x64;

— Microsoft Windows 8, x86-32 і x64;

— Microsoft Windows 7 SP1, x86-32 і x64;

— Ubuntu Linux 10,04 або вище, x86-32 (з встановленими GTK +, libwebkitgtk-1.0-0, libudev, libssl 0.9.8 і вище).

3 ПІДГОТОВКА ДО РОБОТИ

До складу додатки повинні входити файли для його функціонування та залежності, див. рис. 3.1.

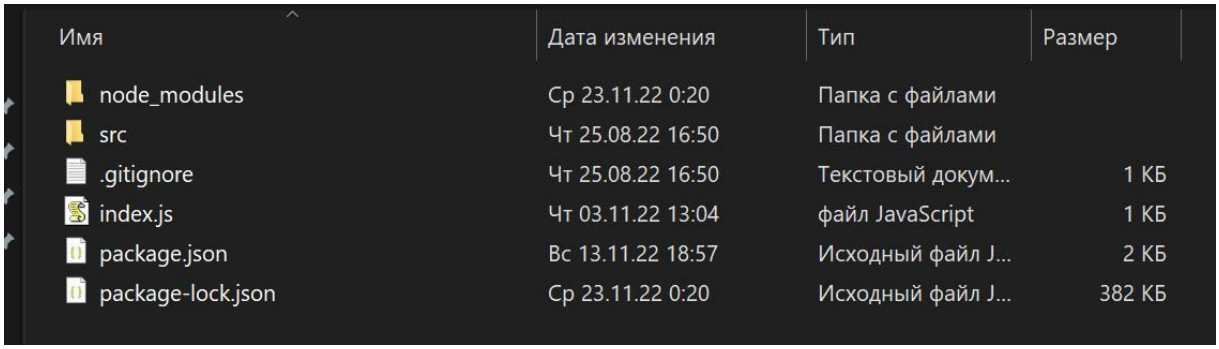


Рисунок 3.1 – Файли додатку

Для початку роботи з програмою, необхідно запустити додаток командою npm run nodemon-dev, див. рис. 3.2.

Рисунок 3.2 – Термінал виклику додатку

Після завантаження необхідних модулів, програма створить локальний сервер за адресом localhost, з портом 3000. Адреса являється точкою входу в програмний додаток.

Для подальшої роботи, необхідно перейти в веб-браузер.

4 ОПИС ОПЕРАЦІЙ

Початок роботи додатку показано на рис. 4.1.

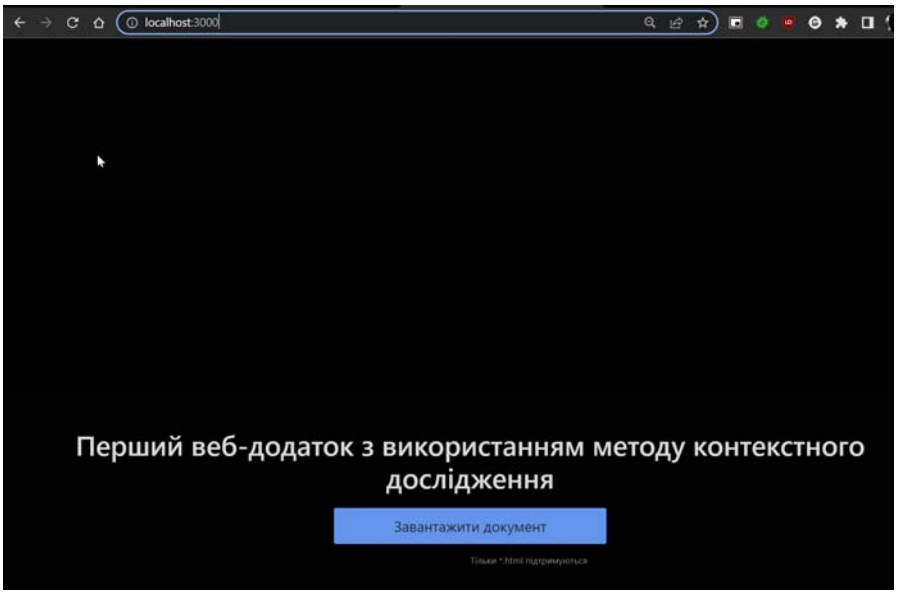


Рисунок 4.1 – Запуск додатку

При запуску з’являється початкове вікно, з текстом до користувача та кнопкою для завантаження гіпертекстових файлів, з пояснювальним підписом.

Для запуску аналізу, необхідно натиснути кнопку «Завантажити документ» та обрати в екранній формі провідника файл з розширенням .html/.html, див. рис. 4.2.

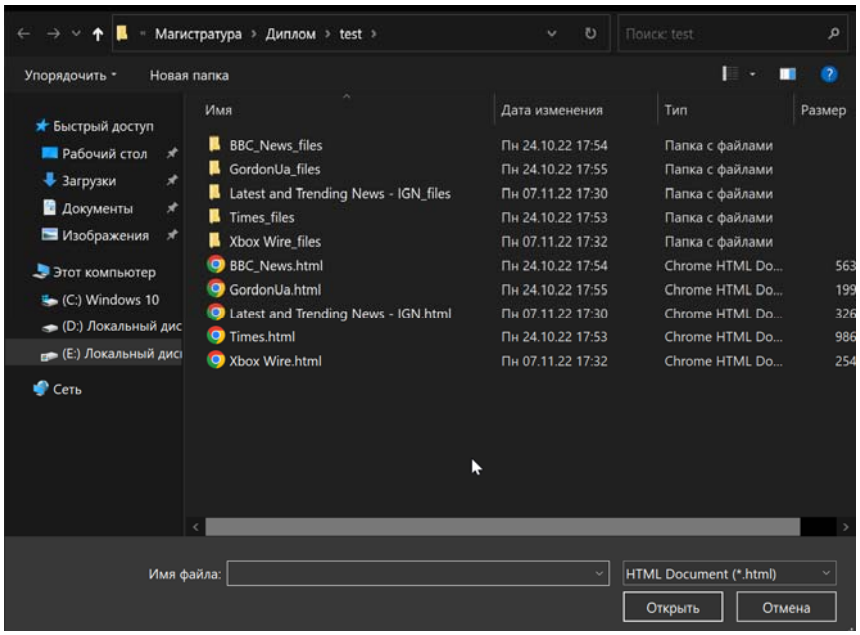


Рисунок 4.2 – Екранна форма провідника системи

Необхідно вибрати файл з зазначеним розширенням та підтвердити свій вибір. Після цього, дочекатись, поки сервер отримає файл та виконає аналіз.

Після завершення аналізу, додаток переводить користувача на сторінку з результатами, див. рис. 4.3.

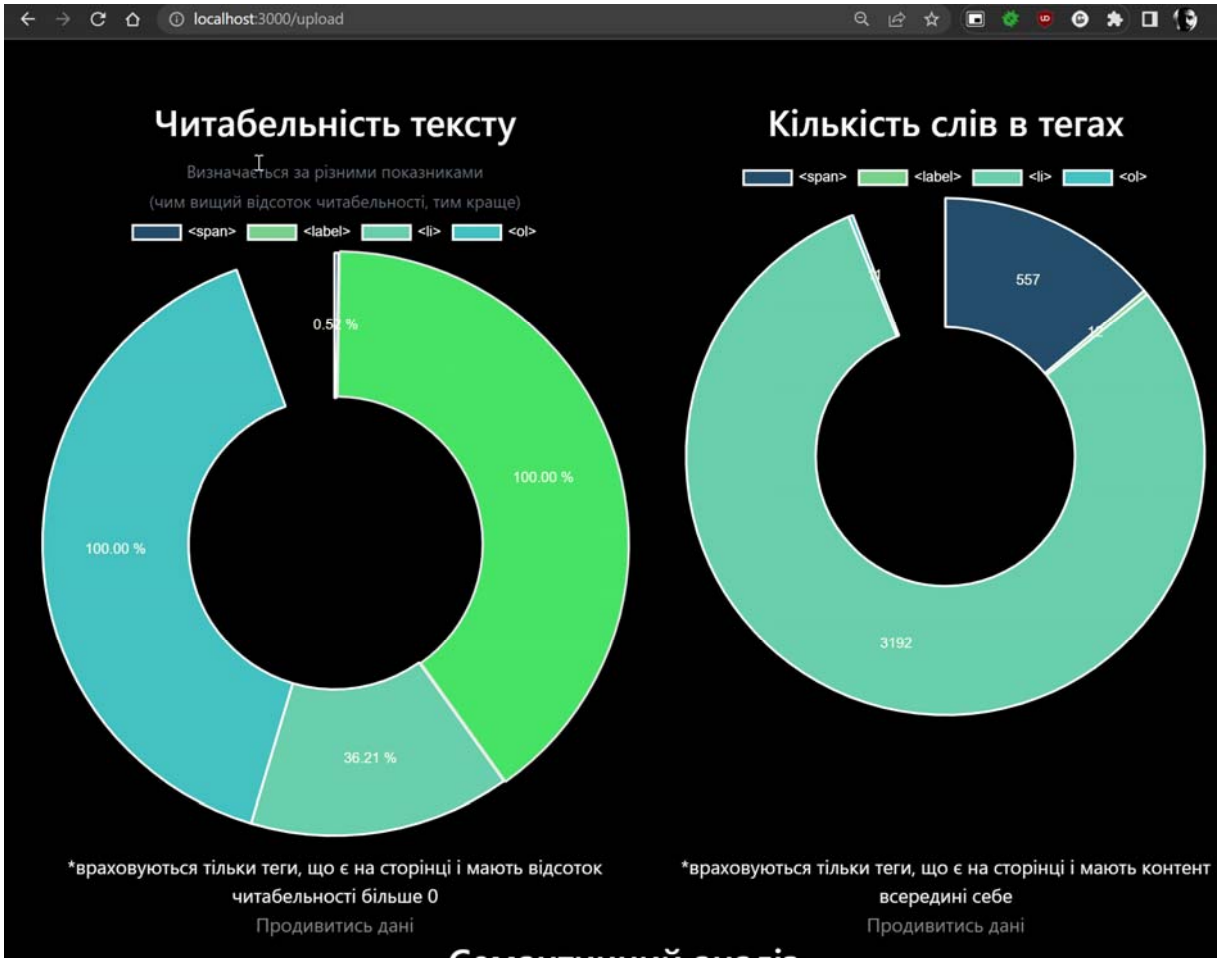


Рисунок 4.3 – Сторінка результатів аналізу

Сторінка містить діаграми з результатами аналізу критеріїв – текстовий та семантичний аналіз.

При натисканні на легенду діаграми, елемент приховується з діаграми, при повторному натисканні – елемент відображається знову, див. рисунок 4.4.

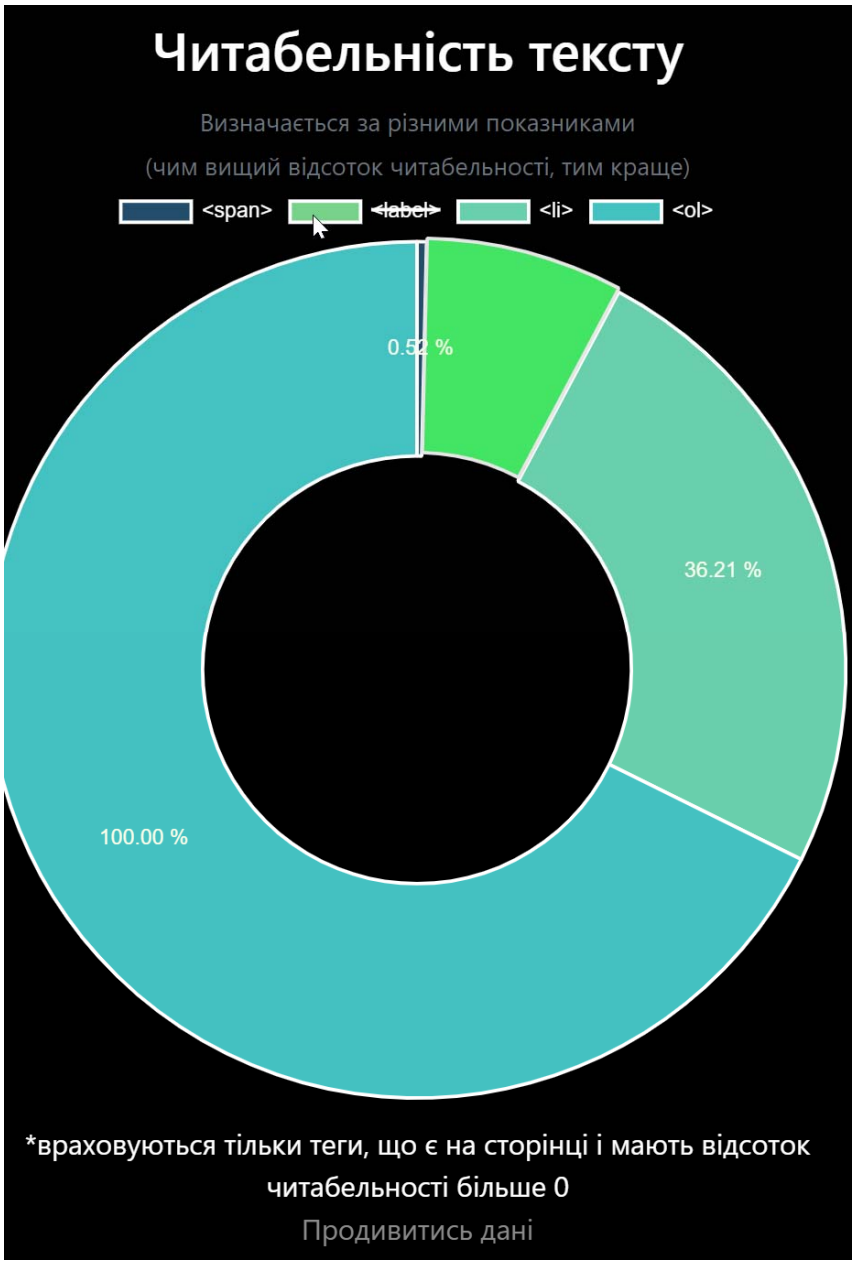


Рисунок 4.4 – Натискання на легенду діаграми

Для перегляду текстового представлення даних, необхідно натиснути на текст «Продивитися дані» – відкриється модальне вікно з текстовим представленням аналізованих даних, див. рисунок 4.5.

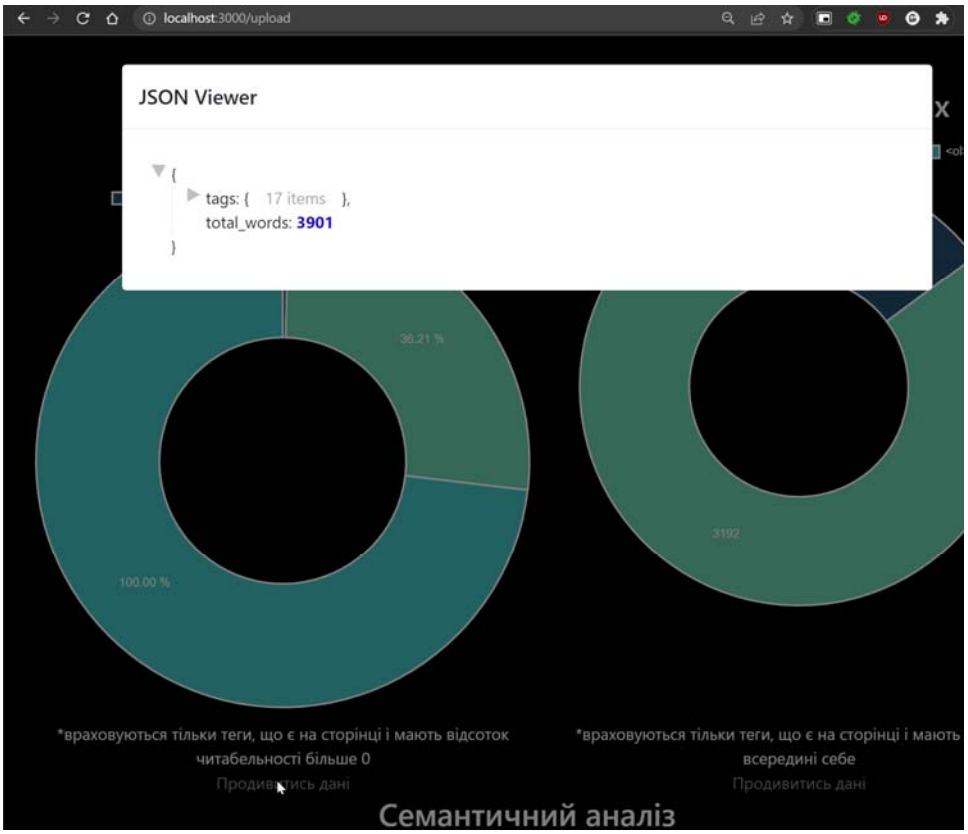


Рисунок 4.5 – Виклик текстового представлення даних

Текстові дані представлені в форматі JSON, мають ієрархічну структуру. Для відкриття або закриття елементу, необхідно натиснути на елемент, див. рис. 4.6.

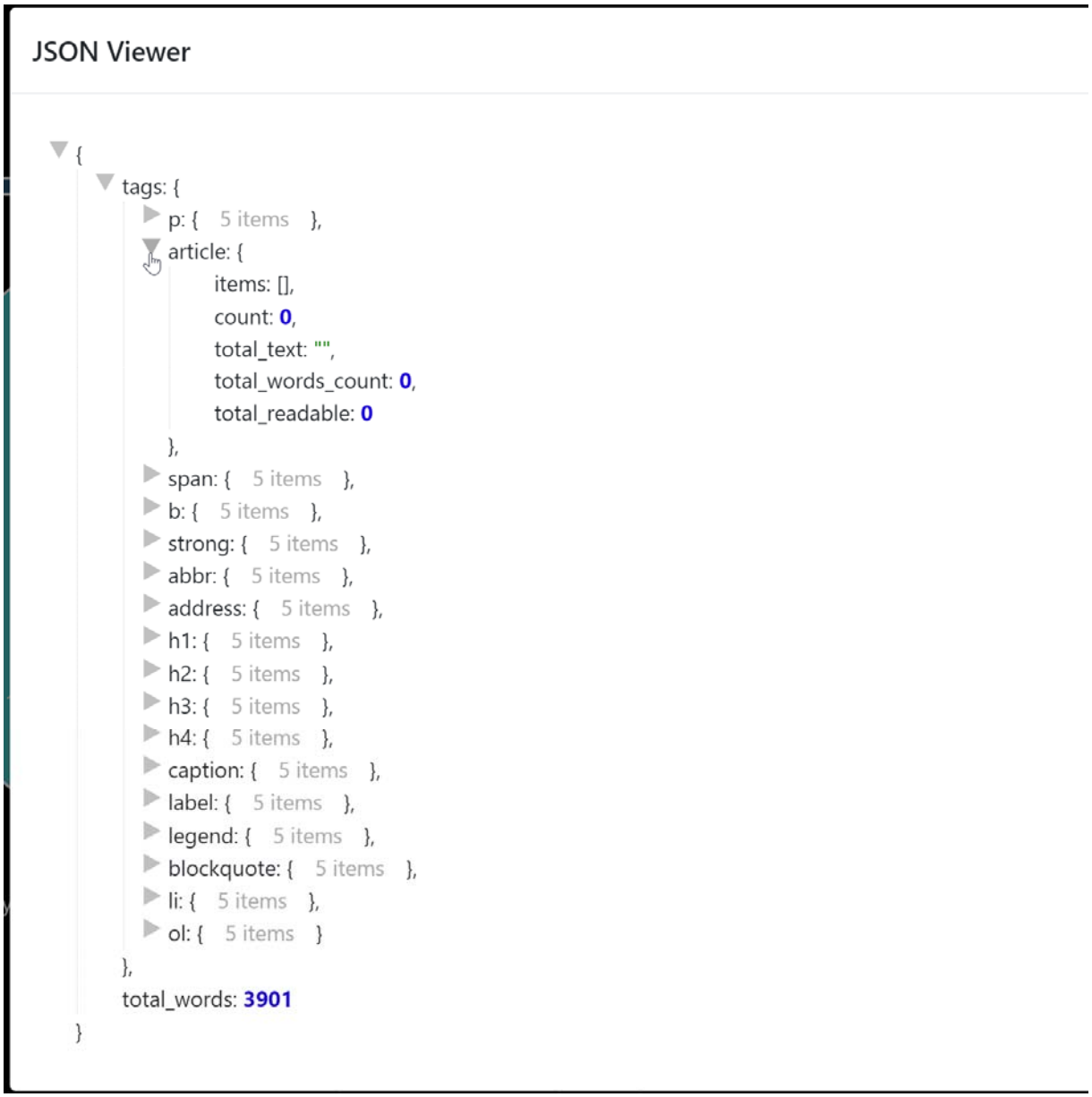


Рисунок 4.6 – Відкриття елементу текстового представлення

Для закриття модального вікна, необхідно натиснути на область за межею модального вікна.

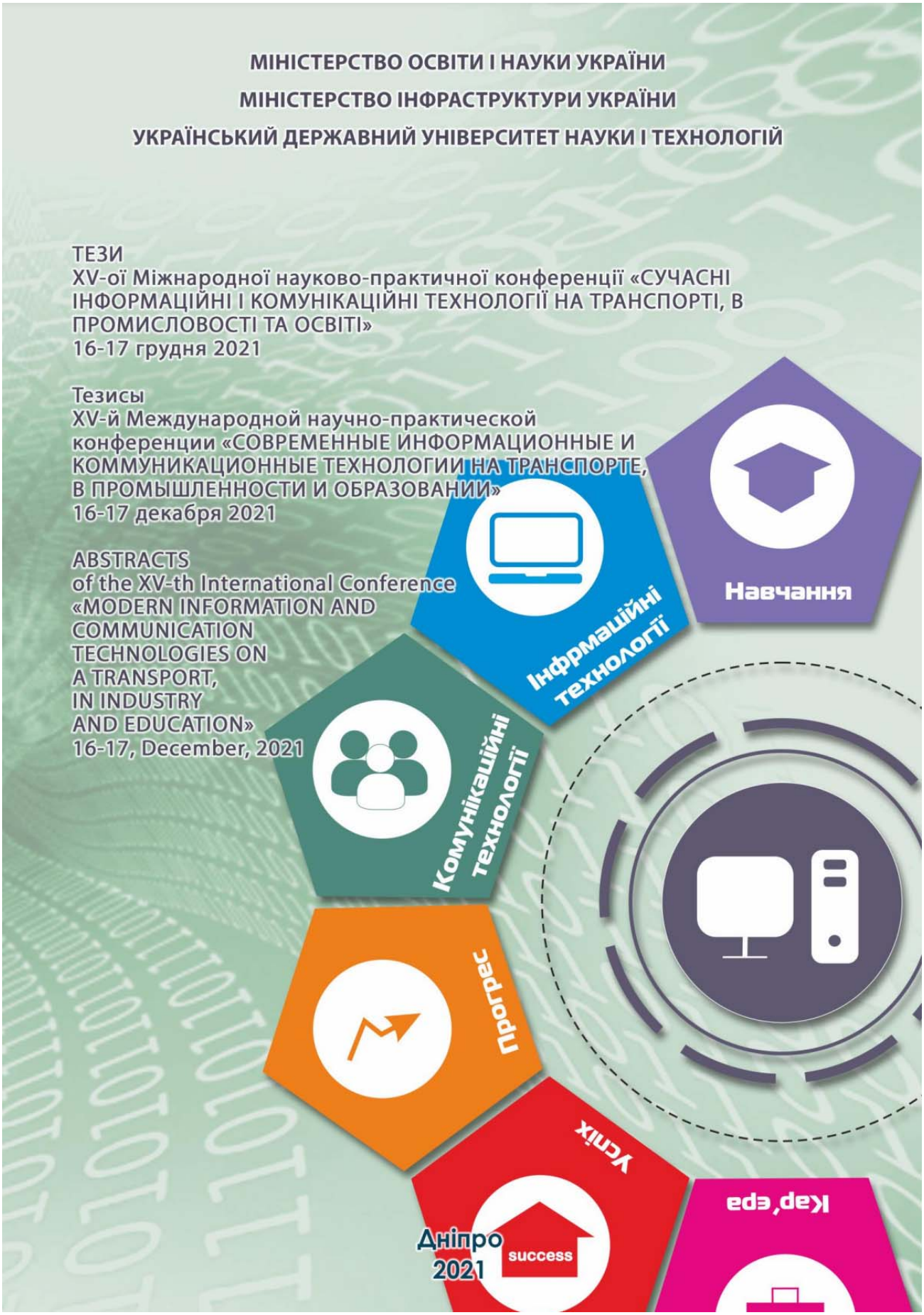
5 АВАРІЙНІ СИТУАЦІЇ

У таблиці 5.1 наведені повідомлення користувачу, що можуть з’явитись у процесі аналізу.

Таблиця 5.1 – Повідомлення користувачу

Текст повідомлення	Опис ситуації	Рекомендовані дії
«Помилка! Файл неправильного формату!»	Користувач намагається завантажити файл, відмінний від .html/.htm	Перевірити формат файлу при завантаженні
«Помилка! Неможливо завантажити файл»	Користувач використовує застарілу версію веб-браузеру	Перевірити актуальну веб-браузеру, оновити веб-браузер
«Помилка! Неможливо перейти за адресом!»	Користувач намагається перейти за адресою localhost:3000, не увімкнувши додаток	Запустити додаток, дочекатись повідомлення про запуск серверу, спробувати знову

Додаток В


МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
МІНІСТЕРСТВО ІНФРАСТРУКТУРИ УКРАЇНИ
УКРАЇНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ НАУКИ І ТЕХНОЛОГІЙ
ТЕЗИ
XV-ої Міжнародної науково-практичної конференції «СУЧАСНІ ІНФОРМАЦІЙНІ І КОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ НА ТРАНСПОРТІ, В ПРОМИСЛОВОСТІ ТА ОСВІТІ»
16-17 грудня 2021
Тезисы
XV-й Международной научно-практической конференции «СОВРЕМЕННЫЕ ИНФОРМАЦИОННЫЕ И КОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ НА ТРАНСПОРТЕ, В ПРОМЫШЛЕННОСТИ И ОБРАЗОВАНИИ»
16-17 декабря 2021
ABSTRACTS
of the XV-th International Conference «MODERN INFORMATION AND COMMUNICATION TECHNOLOGIES ON A TRANSPORT, IN INDUSTRY AND EDUCATION»
16-17, December, 2021
Навчання
Інфрмаційні технології
Комунікаційні технології
Прогрес
Успіх
Дніпро 2021
success
Кар'єра

Міністерство освіти і науки України

Український державний університет науки та технологій

Східний науковий центр транспортної академії наук

TEMPUS: CITISET & SEREIN & CRENG

ТЕЗИ
XV Міжнародної науково-практичної конференції
«СУЧАСНІ ІНФОРМАЦІЙНІ ТА КОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ НА ТРАНСПОРТІ, В ПРОМИСЛОВОСТІ ТА ОСВІТІ»
до 100-річчя професора Євгена Мироновича Шафіта

ABSTRACTS
of the XV International Conference
«MODERN INFORMATION AND COMMUNICATION TECHNOLOGIES ON A TRANSPORT, IN INDUSTRY AND EDUCATION»
for the 100th anniversary of Professor Yevgen Mironovich Shafit

ТЕЗИСЫ
XV Международной научно-практической конференции
«СОВРЕМЕННЫЕ ИНФОРМАЦИОННЫЕ И КОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ НА ТРАНСПОРТЕ, В ПРОМЫШЛЕННОСТИ И ОБРАЗОВАНИИ»
на 100-летие профессора Евгения Мироновича Шафита
16.12.2021 – 17.12.2021

Дніпро
2021

Контекстне дослідження веб-сайтів

Богуцький Д.В., Горбова О.В.,
Український державний університет науки і технологій, Україна

Висока конкуренція в інтернет-середовищі призводить до того, що власникам веб-проектів необхідно постійно генерувати нові ідеї, покращуючи контент, а також впроваджуючи різні трендові «фішки» на свій сайт. В мережі Інтернет налічується велика кількість інформації щодо інструментів для оцінки та відстеження поведінкових факторів на сайті, тому існує задача вивчення цих факторів та їх вплив на ІТ-продукти.

У зв'язку з стрімким поширенням інформаційних технологій, використанням web-ресурсів постає актуальним питання дослідження зручності та практичності використовуваних засобів при проектуванні та створення веб-сайтів, як для розробника, а так і для звичайного користувача у напрямі зручністі експлуатації та простотою використання. Зокрема, задача охоплює такі її складові частини як веб-сайти, що в собі утримують такі елементи для дослідження як web-технології та дизайну, у рамках представленої задачі.

Для вирішення задачі, використовується метод контекстних дослідження, який включає в себе детальне дослідження та інтерв'ю невеликої групи людей за темами предметної області, для отримання реального представлення досліджуваного веб-сайту. Метод контекстних досліджень базується на основних методах:

- врахування контексту в межах обраної задачі, при дослідженні веб-сайту;
- спільна оцінка веб-сайту користувачем та розробником;
- основною метрикою при представлені оцінки веб-сайту представляється зручність його використання користувачем.

Метод контекстних досліджень, на відміну від звичайного опитування чи інтерв'ю, дозволяє уникнути проблем при зборі інформації від користувачів, що не етапі опитування намагались згадати про процес та пояснити його особливості, яким в даний момент часу не займаються.

Ураховуючи особливості, що покладені в основу контекстних досліджень, метод дозволяє уникнути абстрактного представлення предметної області та задачі, в рамках яких він використовується, та отримати чітке представлення результатів дослідження. Контекст дослідження проводиться в природній середі користувача, не утримуючи його в штучних рамках дослідження, для отримання результату. Дослідження проводиться дослідником, задача якого – спостереження за користувачем, що виконує задачу в звичних умовах та ставить йому питання для отримання розуміння того, які задачі виконує користувач, для яких цілей він це робить та який результат сподівається побачити.

Після закінчення роботи над контекстними дослідженнями, які часто застосовуються спільно з аналізом завдань, проводяться загальні збори, для обговорення отриманих висновків та інтерпретувати результати досліджень. Робоча група проводить пошук взаємозв'язків у кількісних даних, що послужить основою створення шаблонів і сценаріїв. Підсумком цього має стати загальне розуміння робочих процесів, моделей мислення та типової поведінки користувачів.

Підводячи підсумок, що при проведенні контекстного дослідження сайтів необхідно звернути увагу на простоту написаного контенту та інтерфейсу, вигідне розміщення інформації, встановити акцент на зображенні, його естетичний зовнішній вигляд та підказки для інтерфейсу. Показники цих метрик дозволять якісно виконати контекстне дослідження та отримати великий масив якісних даних про дії користувачів та отримати інформацію, необхідну для розробки якісних ІТ-продуктів.

ДОДАТОК Г

Міністерство освіти і науки України
Одеський національний технологічний університет
Інститут комп'ютерних систем і технологій
"Індустрія 4.0" ім.П.Н.Платонова

«ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ І АВТОМАТИЗАЦІЯ – 2022»

МАТЕРІАЛИ
XV МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ

20 - 21 ЖОВТНЯ 2022 р.

м.ОДЕСА

УДК 004.588

КОНТЕКСТНЕ ДОСЛІДЖЕННЯ ВЕБ-САЙТІВ

Богуцький Д.В., Горбова О.В
(bohutskiy2@gmail.com, alexandra.gorbova@mail.com)
Український державний університет науки і технологій (Україна)

В тезах розглядається проблема дослідження сучасних веб-сайтів, питання зручності та простота експлуатації веб-продуктів як користувач та способи проектування веб-продуктів як розробника. Описується метод контекстних досліджень для аналізу веб-сайтів, його особливості, проводиться порівняння з іншими методами.

Контекстне дослідження є одним із альтернатив методу еталонного тестування, у якому зручність оцінюється в лабораторних умовах, а чи не у звичної користувача робочої обстановці. При контекстному дослідженні робота, час, мотивація та соціальні фактори, що

впливають на користувача, залишаються такими ж, як у реальному світі, на відміну від лабораторних досліджень, де ці фактори контролюються експериментатором [1].

У зв'язку з стрімким поширенням інформаційних технологій, використанням web-ресурсів постає актуальним питання дослідження зручності та практичності використовуваних засобів при проектуванні та створення веб-сайтів, як для розробника, а так і для звичайного користувача у напрямі зручністі експлуатації та простотою використання. Зокрема, задача охоплює такі її складові частини як веб-сайти, що в собі утримують такі елементи для дослідження як web-технології та дизайну, у рамках представленої задачі.

Для вирішення задачі, використовується метод контекстних дослідження, який включає в себе детальне дослідження та інтерв'ю невеликої групи людей за темами предметної області, для отримання реального представлення досліджуваного веб-сайту. Метод контекстних досліджень базується на основних методах, таких як врахування контексту в межах обраної задачі, при дослідженні веб-сайту, спільна оцінка веб-сайту користувачем та розробником, а також основною метрикою при представлені оцінки веб-сайту представляється зручність його використання користувачем.

Метод контекстних досліджень, на відміну від звичайного опитування чи інтерв'ю, дозволяє уникнути проблем при зборі інформації від користувачів, що не етапі опитування намагались згадати про процес та пояснити його особливості, яким в даний момент часу не займаються.

Ураховуючи особливості, що покладені в основу контекстних досліджень, метод дозволяє уникнути абстрактного представлення предметної області та задачі, в рамках яких він використовується, та отримати чітке представлення результатів дослідження. Контекст дослідження проводиться в природній середі користувача, не утримуючи його в штучних рамках дослідження, для отримання результату. Дослідження проводиться дослідником, задача якого – спостереження за користувачем, що виконує задачу в звичних умовах та ставить йому питання для отримання розуміння того, які задачі виконує користувач, для яких цілей він це робить та який результат сподівається побачити.

Після закінчення роботи над контекстними дослідженнями, які часто застосовуються спільно з аналізом завдань, проводяться загальні збори, для обговорення отриманих висновків та інтерпретувати результати досліджень. Робоча група проводить пошук взаємозв'язків у кількісних даних, що послужить основою створення шаблонів і сценаріїв. Підсумком цього має стати загальне розуміння робочих процесів, моделей мислення та типової поведінки користувачів.

Підводячи підсумок, що при проведенні контекстного дослідження сайтів необхідно звернути увагу на простоту написаного контенту та інтерфейсу, вигідне розміщення інформації, встановити акцент на зображенні, його естетичний зовнішній вигляд та підказки для інтерфейсу. Показники цих метрик дозволять якісно виконати контекстне дослідження та отримати великий масив якісних даних про дії користувачів та отримати інформацію, необхідну для розробки якісних IT-продуктів.

Список використаної літератури

[1] "Контекстное исследование (Contextual Inquiry)." in *Лекции.Ком*. [Online]. Available: https://lektsii.com/2-44192.html