МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ЛУЦКОВ МАКСИМ ПАВЛОВИЧ

Допускається до захисту:
В.о. завідувача кафедри
інформаційних технологій
канд. техн. наук, доцент
_____ Оксана ЗЕЛІНСЬКА
«____»_____ 2024р.

ВЕБ-ДОДАТОК ДЛЯ АНАЛІЗУ ТА ПОКРАЩЕННЯ ФУНКЦІОНАЛЬНОСТІ ВЕБ-САЙТІВ

Спеціальність 122 Комп’ютерні науки
Кваліфікаційна (магістерська) робота

Керівник:
П.В. Римар, старший викладач
кафедри інформаційних технологій
Науковий консультант:
Ю.С. Антонов, доцент кафедри
інформаційних технологій
к.фіз.-мат.наук, доцент

Оцінка: _____ / _____ / _____
(бали/за шкалою ЄКТС/за національною шкалою)
Голова ЕК: _____

Вінниця – 2024

АНОТАЦІЯ

Луцков М.П. Веб-додаток для аналізу та покращення функціональності веб-сайтів. Спеціальність 122 «Комп’ютерні науки», освітня програма «Комп’ютерні технології обробки даних». Донецький національний університет імені Василя Стуса, Вінниця, 2024.

Робота присвячена розробці та реалізації веб-додатку, який дозволяє аналізувати та покращувати функціональність веб-сайтів. Основною метою дослідження є створення інструменту, який надає можливість власникам веб-сайтів збирати дані про роботу своїх проектів та отримувати рекомендації щодо їх покращення.

У роботі буде проведено аналіз існуючих підходів до аналізу веб-сайтів та розглянуті інструменти, які можуть використовуватися для цієї мети. Далі буде описано розробку та реалізацію веб-додатку, який включатиме збір даних про роботу веб-сайтів, визначення проблем та надання рекомендацій щодо їх вирішення, а також візуалізацію результатів аналізу.

Ключові слова: веб-додаток, аналіз сайтів, рекомендації.

Робота містить 68 сторінок, 57 рисунків та список літератури з 29 джерелами.

ABSTRACT

Lutskov M.P. A web application for analyzing and improving the functionality of website. Specialty 122 «Computer Science», Educational program «Data Science», Vasyl’ Stus Donetsk National University, Vinnytsia, 2024.

The master's thesis is devoted to the development and implementation of a web application that allows analyzing and improving the functionality of websites. The main goal of the research is to create a tool that provides website owners with the ability to collect data on the performance of their projects and receive recommendations for their improvement.

The work will analyze existing approaches to website analysis and consider tools that can be used for this purpose. Next, the development and implementation of a web

application will be described, which will include data collection on the performance of websites, identification of problems, and the provision of recommendations for their resolution, as well as visualization of the analysis results.

Keywords: web application, site analysis, recommendations.

The work contains 68 pages, 57 figures and a bibliography with 29 sources.

ЗМІСТ

ВСТУП

В сучасному цифровому світі веб-сайти стали невід'ємною складовою частиною бізнесу, комунікації та інформаційного простору. Вони використовуються для представлення товарів і послуг, обміну інформацією, спілкування з користувачами та багато іншого. Однак, успіх веб-сайту значною мірою залежить від його функціональності - здатності задовольнити потреби користувачів та забезпечити зручний та ефективний взаємодію.

Метою даної роботи є аналіз та покращення функціональності веб-сайтів. В рамках практичного досвіду ми вивчали методи та стратегії, що допомагають зрозуміти проблеми, з якими стикаються веб-сайти, та розробити ефективні рішення для їх покращення.

Завдання практичної підготовки включали вибір конкретного веб-сайту для аналізу його функціональності, виявлення проблем та недоліків, розробку плану покращень та їх впровадження. Ми використовували широкий спектр методів та підходів, включаючи аналіз вимог користувачів, тестування, взаємодію з розробниками.

Аналіз та покращення функціональності веб-сайтів є важливим завданням, яке може сприяти поліпшенню користувацького досвіду, збільшенню ефективності бізнесу та забезпеченню конкурентоспроможності в онлайн-середовищі. Під час практичної підготовки ми досліджували цю проблематику та намагалися знайти оптимальні рішення для покращення функціональності веб-сайтів.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОГЛЯД КОНКУРЕНТІВ

1.1. Опис актуальності предметної області

Аналіз та покращення функціональності веб-сайтів є актуальною темою в сучасному цифровому середовищі. Ось кілька ключових аспектів, які підкреслюють актуальність даної теми:

1. Зростання конкуренції: З кожним роком кількість веб-сайтів зростає експоненційно, і конкуренція за увагу користувачів стає все більш жорсткою. Веб-сайти, які пропонують високу функціональність та забезпечують зручну взаємодію з користувачами, мають більші шанси здобути перевагу над конкурентами.

2. Зміни в поведінці користувачів: Користувачі стають все вимогливішими та очікують швидкого та зручного доступу до інформації. Вони швидко відходять від веб-сайтів, які не задовольняють їхні потреби або мають низьку функціональність. Врахування потреб та очікувань користувачів і розробка веб-сайтів з високою функціональністю стають важливими для залучення та утримання аудиторії.

3. Вплив на бізнес: Веб-сайт є важливим інструментом для бізнесу, оскільки він дозволяє представляти продукти та послуги, залучати клієнтів, встановлювати контакт з ними та здійснювати торгівлю в онлайн-режимі. Оптимізація функціональності веб-сайту може покращити конверсію, збільшити продажі та сприяти розвитку бізнесу.

4. Розвиток технологій: Швидкий розвиток технологій та постійні зміни у веб-середовищі вимагають постійного оновлення та покращення функціональності веб-сайтів. Нові можливості, такі як штучний інтелект, мобільність, віртуальна та доповнена реальність, створюють нові виклики та можливості для розширення функціональності веб-сайтів.

5. Вплив на користувачів: Функціональність веб-сайту має прямий вплив на користувачів, їхню задоволеність та взаємодію з сайтом. Веб-сайти з низькою

функціональністю можуть призводити до незадоволення, збитку для репутації та втрати клієнтів. Забезпечення високої функціональності веб-сайту дозволяє поліпшити користувацький досвід, залучити та утримувати більше користувачів.

Оглядаючи ці аспекти, очевидно, що аналіз та покращення функціональності веб-сайтів є важливим завданням для підтримки ефективної присутності в онлайн-середовищі. Розуміння актуальності цієї теми дозволяє нам усвідомити важливість наших досліджень та дій в цьому напрямку.

1.2. Визначення веб-сайту та його функціональності

Веб-сайт - це колекція веб-сторінок, що знаходяться на інтернеті та доступні за допомогою веб-переглядачів. Веб-сайти використовуються для різноманітних цілей, включаючи надання інформації, представлення продуктів та послуг, комунікації з користувачами та багато іншого. Вони можуть бути статичними, коли вміст не змінюється часто, або динамічними, коли вміст оновлюється регулярно та змінюється в залежності від потреб користувачів.

Функціональність веб-сайту визначається набором можливостей та функцій, які він пропонує користувачам. Це охоплює різноманітні аспекти, включаючи навігацію, розташування та взаємодію зі сторінками, функції пошуку, реєстрації та аутентифікації користувачів, роботу з формами, зберігання та обробку даних, інтеграцію з іншими системами та багато іншого. Функціональність веб-сайту має на меті забезпечити зручність використання та задоволення потреб користувачів.

Основні складові функціональності веб-сайту включають:

1. Навігація: Веб-сайт повинен мати чітку та логічну структуру, що дозволяє користувачам швидко та легко знаходити необхідну інформацію. Це включає меню, посилання та інші елементи, які допомагають навігувати по сайту.

2. Взаємодія з користувачем: Веб-сайт повинен забезпечувати можливості взаємодії з користувачами, такі як заповнення форм, коментування, оцінювання, спілкування через чат або форум, підписка на розсилки тощо.

3. Розташування та візуальний дизайн: Ефективний розміщення елементів на сторінці, зручність читання, використання привабливого дизайну та графіки сприяють залученню користувачів та поліпшенню їх взаємодії з веб-сайтом.

4. Пошук та фільтрація: Можливість швидкого та точного пошуку інформації на веб-сайті, а також можливість використання фільтрів для звуження результатів, дозволяють користувачам знайти потрібну інформацію швидко та ефективно.

5. Адаптивність: Веб-сайт повинен бути адаптивним до різних пристроїв, таких як комп'ютери, смартфони, планшети. Це дозволяє користувачам зручно переглядати та використовувати веб-сайт на будь-якому пристрої.

6. Швидкість та продуктивність: Веб-сайт повинен працювати швидко та ефективно, забезпечуючи мінімальний час завантаження сторінок та реагуючи на запити користувачів миттєво.

1.3 Огляд основних методів та підходів до аналізу та покращення функціональності веб-сайтів

Аналіз та покращення функціональності веб-сайтів вимагають систематичного підходу та використання різноманітних методів і інструментів. Ось огляд деяких основних методів та підходів, які застосовуються у цьому процесі:

Аналіз поведінки користувачів: Цей підхід базується на зборі та аналізі даних про поведінку користувачів на веб-сайті. Використовуються інструменти веб-аналітики, які надають інформацію про кількість відвідувачів, популярність різних сторінок, шляхи навігації та інші метрики. Це дозволяє виявити проблемні або неефективні елементи веб-сайту та визначити напрямки для його покращення.

Тестування користувацького досвіду (User Experience Testing): Цей метод включає проведення спеціальних тестів із залученням користувачів для оцінки їх взаємодії з веб-сайтом. Це може включати проведення фокус-груп, інтерв'ю з користувачами, створення прототипів та тестування їх на реальних

користувачах. Результати тестування допомагають ідентифікувати проблеми, зрозуміти потреби та пріоритети користувачів та внести відповідні зміни до веб-сайту.

Аналіз конкурентів: Цей підхід полягає в дослідженні та аналізі веб-сайтів конкурентів для виявлення їх сильних та слабких сторін. Спостереження за конкурентами допомагає отримати ідеї та інсайти щодо можливих покращень функціональності. Це може включати аналіз навігації, структури сторінок, функціональних можливостей, швидкості завантаження та інших аспектів.

Збір фідбеку від користувачів: Важливим елементом аналізу та покращення функціональності є залучення фідбеку від реальних користувачів. Це може бути здійснено через форми зворотного зв'язку на веб-сайті, опитування, коментарі, соціальні медіа та інші канали. Збирання думок та пропозицій користувачів допомагає виявити їх потреби та проблеми, а також знайти способи покращення функціональності.

Експертний аналіз: Цей підхід включає залучення фахівців у галузі веб-дизайну, веб-розробки та маркетингу для проведення огляду веб-сайту та його функціональності. Експерти виявляють проблемні аспекти, роблять рекомендації щодо оптимізації, вдосконалення функціоналу та покращення користувацького досвіду.

Усі ці методи та підходи можуть бути використані окремо або в поєднанні один з одним для аналізу та покращення функціональності веб-сайтів. Важливо регулярно оновлювати та покращувати веб-сайт, забезпечуючи його ефективну роботу та задоволення потреб користувачів.

1.4 Google Analytics

Google Analytics - це безкоштовний інструмент веб-аналітики, розроблений компанією Google, який надає широкі можливості для аналізу та вимірювання різних аспектів функціональності веб-сайтів. Він дозволяє збирати, аналізувати та вивчати дані про відвідуваність та поведінку користувачів на веб-сайті з метою виявлення проблем та вдосконалення його функціональності.

Основні можливості Google Analytics включають:

1. Аналітика відвідуваності: Google Analytics надає детальну інформацію про кількість відвідувачів, відвідані сторінки, тривалість сеансів та інші метрики. Це дозволяє зрозуміти, як користувачі знаходять та взаємодіють з веб-сайтом.

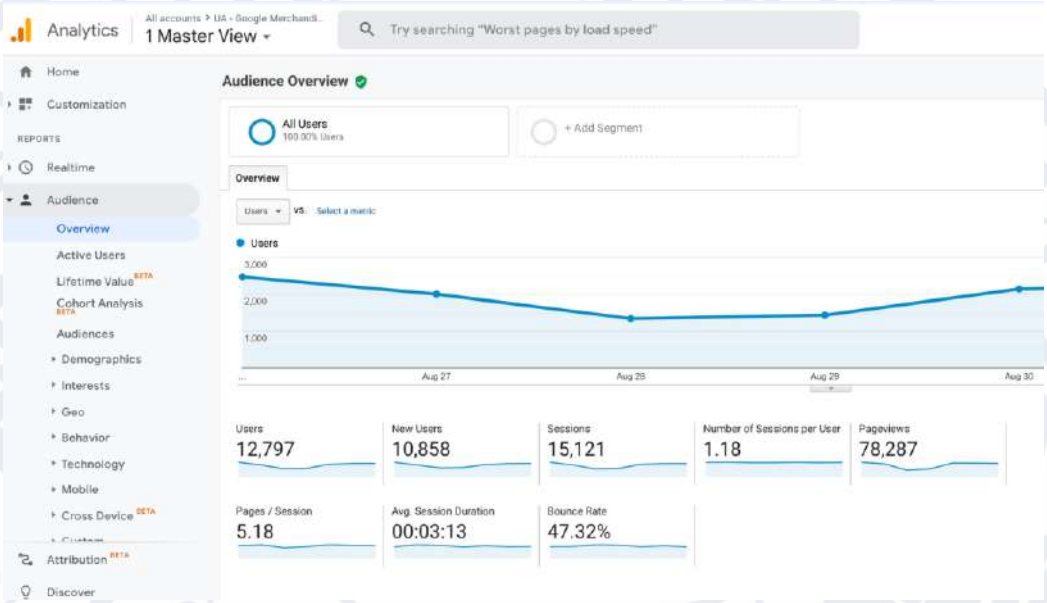


Рисунок 1.1 – Аналітика відвідуванності

2. Аналітика джерел трафіку: Google Analytics показує, з яких джерел користувачі приходять на веб-сайт, таких як пошукові системи, соціальні мережі, рекламні кампанії та інші джерела. Це дозволяє оцінити ефективність різних маркетингових каналів та прийняти рішення щодо їх використання.

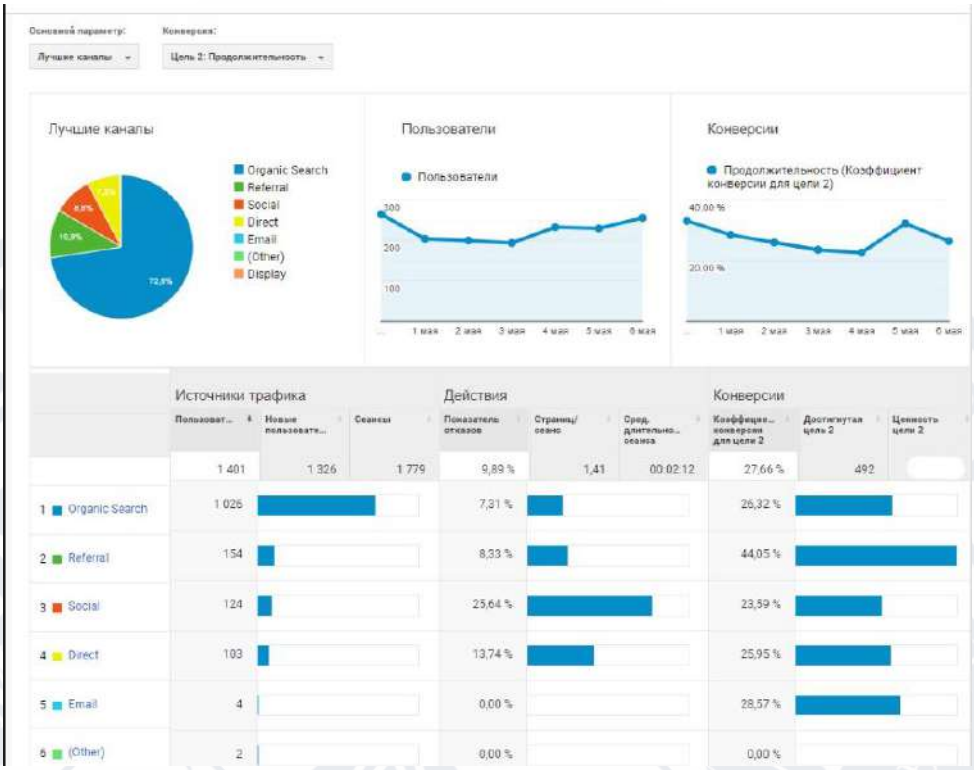


Рисунок 1.2 – Аналіз джерел трафіку

3. Аналітика поведінки користувачів: Google Analytics надає інформацію про шляхи навігації користувачів на веб-сайті, показники відхилень, частоту відвідування сторінок та інші показники поведінки. Це допомагає виявити проблемні або малоефективні елементи веб-сайту та вжити заходів для їх покращення.

Рисунок 1.3 – Аналітика поведінки користувачів

4. Відстеження конверсій: Google Analytics дозволяє встановлювати та відстежувати цілі на веб-сайті, такі як заповнення форми, покупки товарів або завантаження файлів. Це дозволяє вимірювати ефективність веб-сайту в контексті досягнення бізнес-цілей.

5. Створення звітів та налаштування сповіщень: Google Analytics надає можливість створювати різноманітні звіти, графіки та настроювати сповіщення про важливі події або показники. Це дозволяє зосередитися на ключових метриках та швидко виявляти зміни у функціональності веб-сайту.

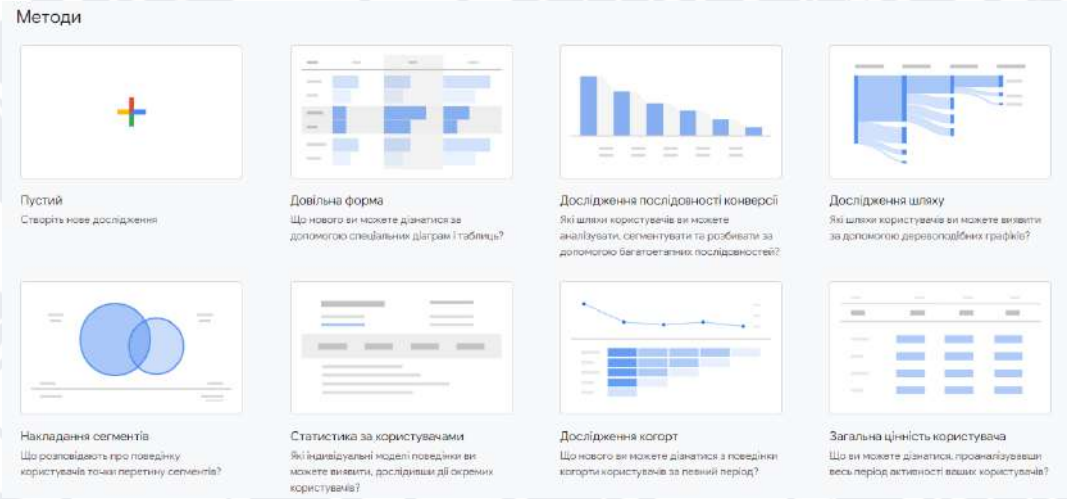


Рисунок 1.4 – Створення звітів

Завдяки широкому спектру функцій та зручному інтерфейсу, Google Analytics є потужним інструментом для аналізу та покращення функціональності веб-сайтів. Він допомагає зрозуміти поведінку користувачів, виявити проблемні аспекти та прийняти обґрунтовані рішення щодо оптимізації та покращення функціональності веб-сайту.

1.5 Hotjar

Hotjar - це аналітичний інструмент, який дозволяє веб-розробникам, маркетологам та дизайнерам отримувати інсайти щодо поведінки користувачів на своєму веб-сайті. Він надає набір інструментів для збору даних про взаємодію користувачів з веб-сторінками, аналізу цих даних і надання звітів, які

допомагають зрозуміти, як користувачі взаємодіють з веб-сайтом та які аспекти можуть бути покращені.

Основні функції та можливості Hotjar включають:

1. Карти тепла (Heatmaps): Вони відображають, як користувачі наводять курсор миші на сторінці, клікають на посилання та прокручують її. Це дає вам уявлення про гарячі та холодні області на сторінці, де користувачі найбільше взаємодіють або не взаємодіють.

Рисунок 1.5 – Карта тепла на сайті

2. Записи сесій (Session Recordings): Hotjar дозволяє записувати сесії користувачів на вашому веб-сайті, включаючи їх взаємодію зі сторінками, наведенням миші, кліками та прокруткою. Це дозволяє переглядати сеанси користувачів у реальному часі, щоб отримати більш глибоке розуміння їхньої поведінки та перспективи.

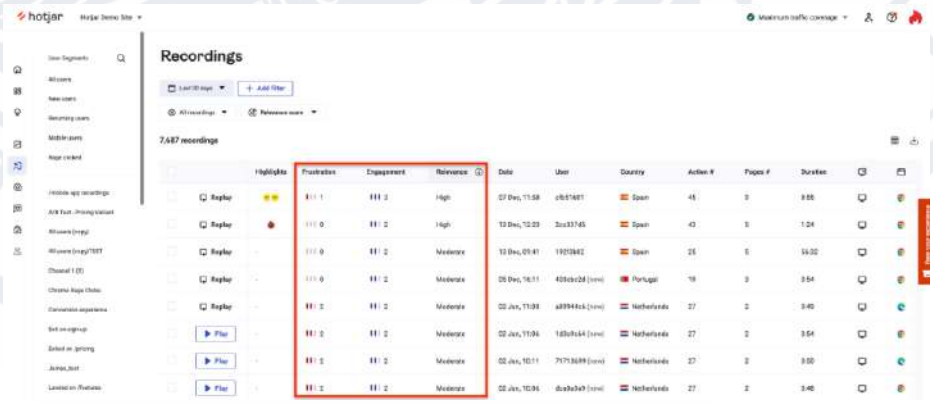


Рисунок 1.6 – Приклад записів сесій

3. Опитування та зворотній зв'язок (Surveys & Feedback): Ви можете створювати поп-ап вікна опитувань або панелі зворотного зв'язку для збору думок та вражень від ваших користувачів. Це допомагає отримати безпосередній зв'язок з вашою аудиторією та з'ясувати, що потрібно поліпшити на вашому веб-сайті.

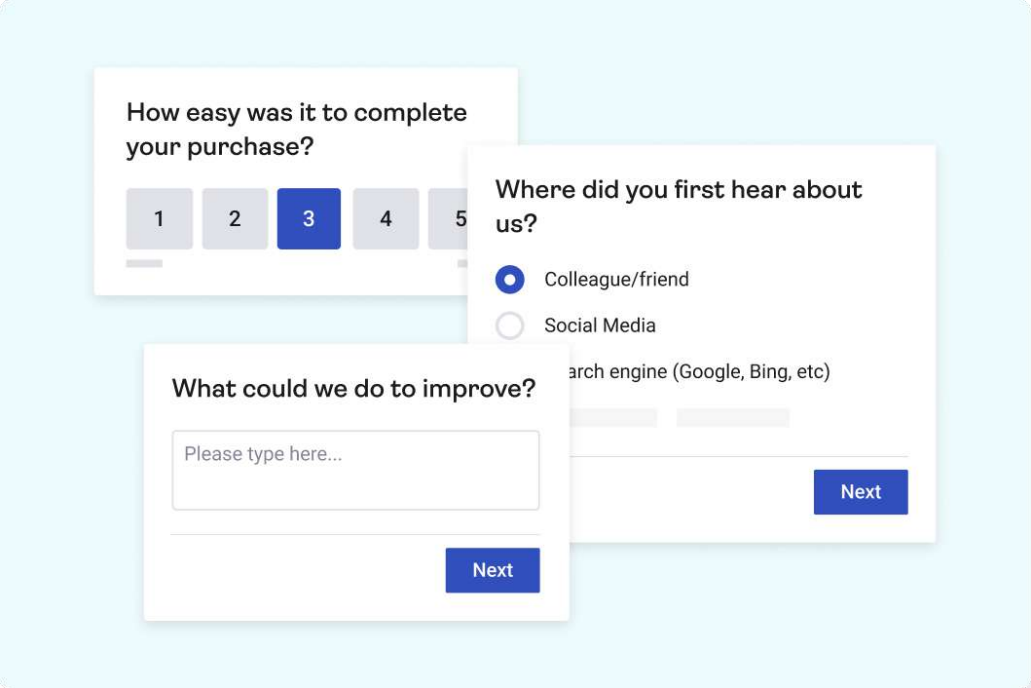


Рисунок 1.7 – Форма зворотного зв’язку

4. Воронки конверсій (Conversion Funnels): Hotjar дозволяє стежити за кроками, які користувачі здійснюють на вашому веб-сайті, від відвідування до досягнення конкретної цілі (наприклад, покупки або заповнення форми). Це дозволяє вам ідентифікувати слабкі місця у воронці конверсій та вживати заходів для їх вдосконалення.

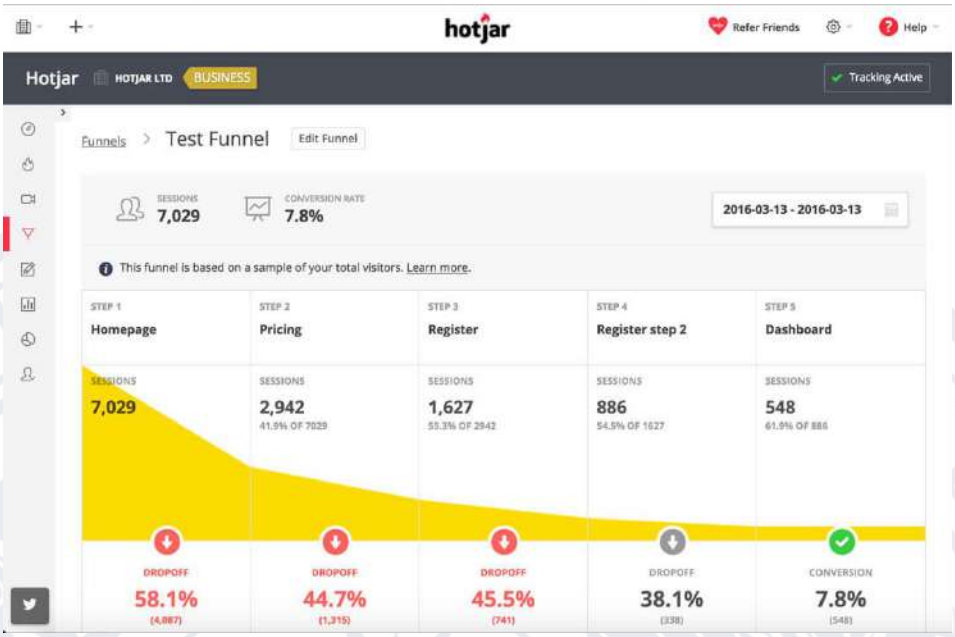


Рисунок 1.8 – Воронки конверсій

5. Огляд форм (Form Analysis): Ви можете вивчати взаємодію користувачів з формами на вашому веб-сайті, включаючи заповнення полів та відправку форм. Це допомагає зрозуміти, наскільки ефективними є ваші форми та як їх можна поліпшити, щоб збільшити конверсію.

Hotjar надає можливості аналізу веб-аналітики, які допомагають збільшити розуміння поведінки користувачів на вашому веб-сайті. Використання цього інструменту може допомогти вам приймати кращі рішення з покращення веб-дизайну, взаємодії та конверсії, що веде до покращення користувацького досвіду та досягнення вашої мети.

1.6 UsabilityHub

UsabilityHub - це онлайн-платформа для проведення тестування користувачами та отримання зворотного зв'язку щодо веб-дизайну, інтерфейсів та інших аспектів веб-продуктів. Вона надає широкий спектр інструментів для збору даних та оцінки користувацького досвіду з метою покращення продукту.

Основні можливості та інструменти, які надає UsabilityHub, включають:

1. Тести першого враження (First Impression Tests): Дозволяють користувачам швидко оцінити веб-сторінку або дизайн, надаючи свої перші

враження та враження, які вони отримують при першому знайомстві з продуктом.

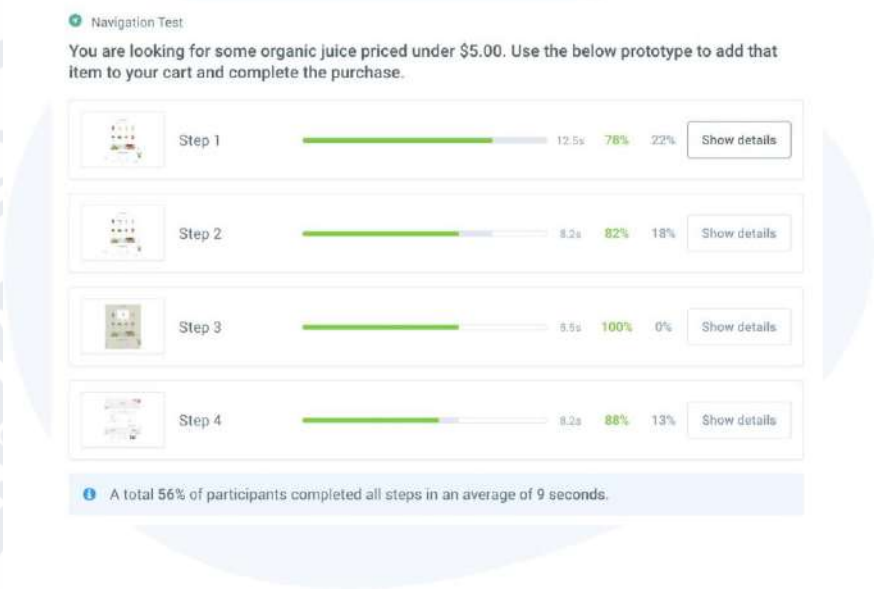


Рисунок 1.9 – Тест першого враження

2. Тести на розташування (Preference Tests): Допомагають визначити, який варіант дизайну або макету є більш привабливим або ефективним за допомогою порівняння різних варіантів і вибору найпопулярнішого.

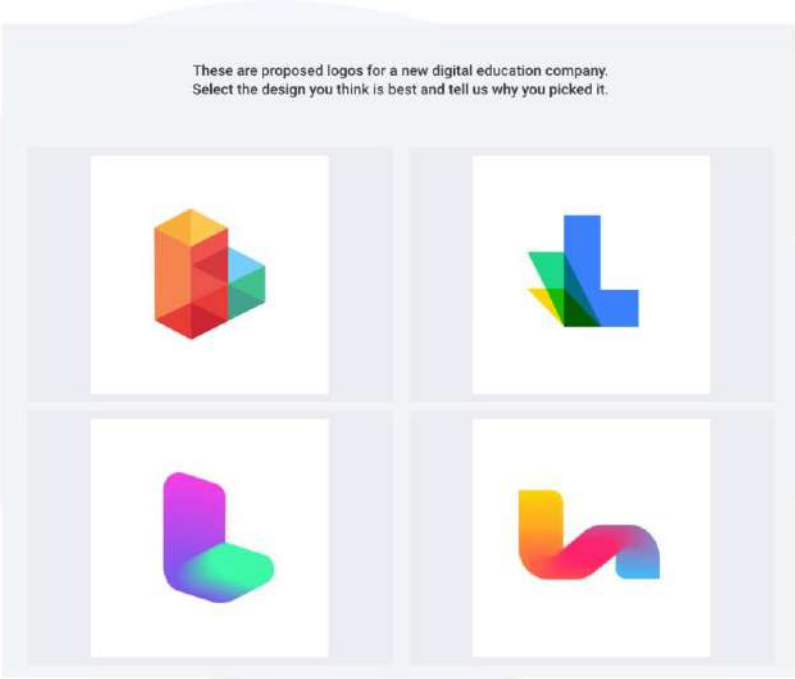


Рисунок 1.10 – Тести на розташування

3. Задачі користувача (User Tasks): Дозволяють створювати сценарії та завдання, які користувачам потрібно виконати на веб-сайті або в додатку. Це допомагає оцінити ефективність інтерфейсу та зрозуміти, наскільки легко користувачам виконувати потрібні дії.

Рисунок 1.11 – Приклад задач користувача

4. Тести на запам'ятовування (Memory Tests): Дозволяють перевірити, наскільки добре користувачі запам'ятовують певну інформацію або елементи дизайну після короткого перегляду. Це може бути корисно для оцінки зрозумілості та запам'ятовуваності важливих елементів.

Рисунок 1.12 – Принцип тесту на запм’ятовування

5. Карти тепла кліків (Click Heatmaps): Надають візуальне відображення того, де користувачі натискають на веб-сторінці. Це допомагає виявити гарячі та холодні зони взаємодії та розуміти, як користувачі взаємодіють з елементами на сторінці.

UsabilityHub надає можливість отримувати цінні відгуки та дані від реальних користувачів, що дозволяє розробникам та дизайнерам отримувати об'єктивну оцінку користувацького досвіду та вносити необхідні покращення до своїх продуктів.

Висновок до 1 розділу

У першому розділі було проведено аналіз предметної області, пов'язаної з аналізом та покращенням функціональності веб-сайтів. Визначено актуальність даної теми, оскільки веб-сайти стають все більш важливими для бізнесу та користувачів.

Було надано визначення веб-сайту та його функціональності, що дозволяє зрозуміти основні аспекти роботи та можливості веб-сайту. Також був проведений огляд основних методів та підходів до аналізу та покращення функціональності веб-сайтів, що дозволяє виявити різні способи оптимізації та

покращення веб-сайтів з метою поліпшення користувацького досвіду. Були розглянуті такі ресурси, як Google Analytics, Hotjar та UsabilityHub, які надають інструменти для збору даних, аналізу та впровадження покращень на веб-сайтах.

Далі було розглянуто різні аспекти покращення функціональності веб-сайту. Особлива увага була приділена оптимізації швидкості завантаження, яка є важливою для забезпечення швидкого та зручного взаємодії користувачів з веб-сайтом. Також було розглянуто впровадження персоналізації як спосіб створення індивідуального досвіду для користувачів.

Крім того, було наголошено на важливості використання наукових та математичних методів при аналізі та покращенні функціональності веб-сайтів. Вони дозволяють зробити обґрунтовані висновки, визначити ефективні стратегії та забезпечити оптимальний досвід користувача.

РОЗДІЛ 2
ОПИС ВИКОРИСТАНИХ ІНСТРУМЕНТІВ ТА ТЕХНОЛОГІЙ РОЗРОБКИ

2.1 Інструменти та утиліти для пришвидшення роботи

Web-програмування є одним з найбільш популярних серед розробників напрямком. Відповідно є великий попит на програмне забезпечення для роботи з веб-сайтами. Різні компанії пропонують свої рішення програмного забезпечення, яке прискорює роботу розробника, завдяки своєму розширеному функціоналу. Тому ми розберемо основні IDE для розробки веб-додатків.

Sublime Text

Sublime Text – це дуже продуктивний текстовий редактор, який був розроблений у 2008 році для платформи Windows. За час свого існування цей редактор встиг стати вкрай популярним серед програмістів та редакторів коду. Починаючи з версії Sublime Text 2, була додана підтримка і для інших операційних систем, таких як Linux і MacOS, що розширило його доступність та користувацьку базу.

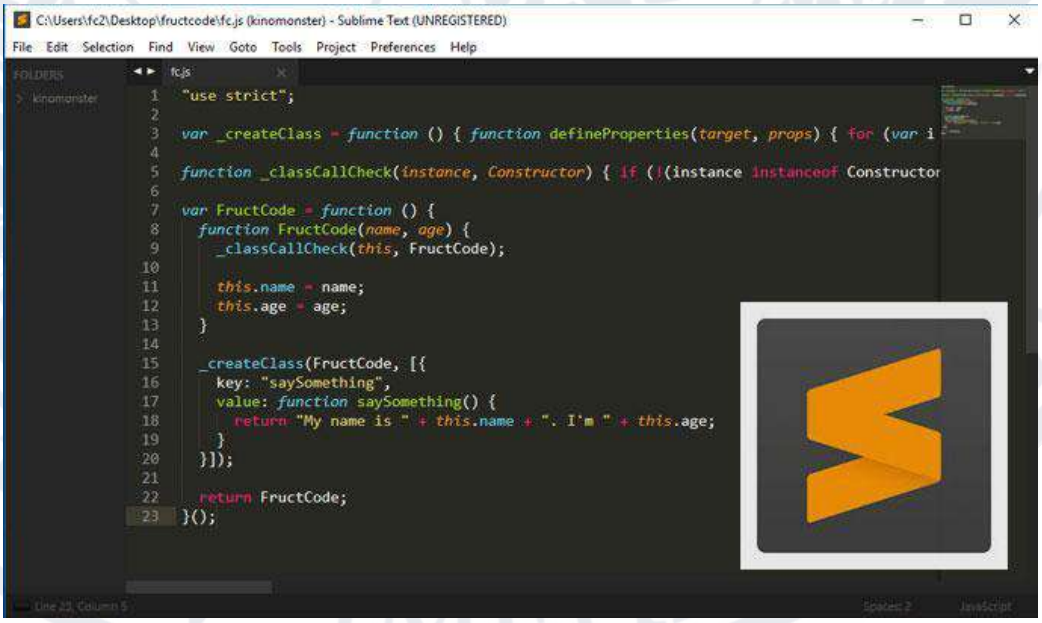


Рисунок 2.1 – Приклад JavaScript коду в Sublime Text 4

У 2013 році вийшла третя версія Sublime Text, яка принесла значні поліпшення та розширення функціоналу. Важливо відзначити, що, незважаючи на те, що попередні версії цього редактора були комерційними, розробники вирішили перейти на безкоштовну модель, що робить його ще більш доступним та зручним для різних категорій користувачів.

Серед основних переваг Sublime Text можна відзначити його компактний розмір програми та вражаючу швидкість роботи. До цього додається добре реалізована функція «емітування», зручна підсвітка синтаксису та зручне дерево файлів вашого проекту.

Sublime Text ідеально підходить для роботи над невеликими проектами та для навчання програмуванню. Крім того, він ефективно працює навіть на повільних комп'ютерах, що дозволяє програмувати без необхідності потужного обладнання.

Звісно, є кілька обмежень у Sublime Text, які роблять його менш ідеальним для великих та складних проектів. Він не надає всіх необхідних плагінів для роботи над великими проектами та не забезпечує компіляцію кількох файлів через термінал. Паралельне відкриття терміналів також не є його сильною стороною, що може призвести до уповільнення роботи з кодом.

Також важливо відзначити, що хоча Sublime Text має плагіни, вони не завжди працюють ідеально і можуть вимагати додаткових налаштувань для досягнення бажаних результатів.

Звісно, Sublime Text – зручний і простий текстовий редактор для роботи з кодом на різних мовах програмування, особливо при розробці веб-сайтів. Він підходить для невеликих та менш складних проектів, і його безкоштовність робить його привабливим для багатьох користувачів.

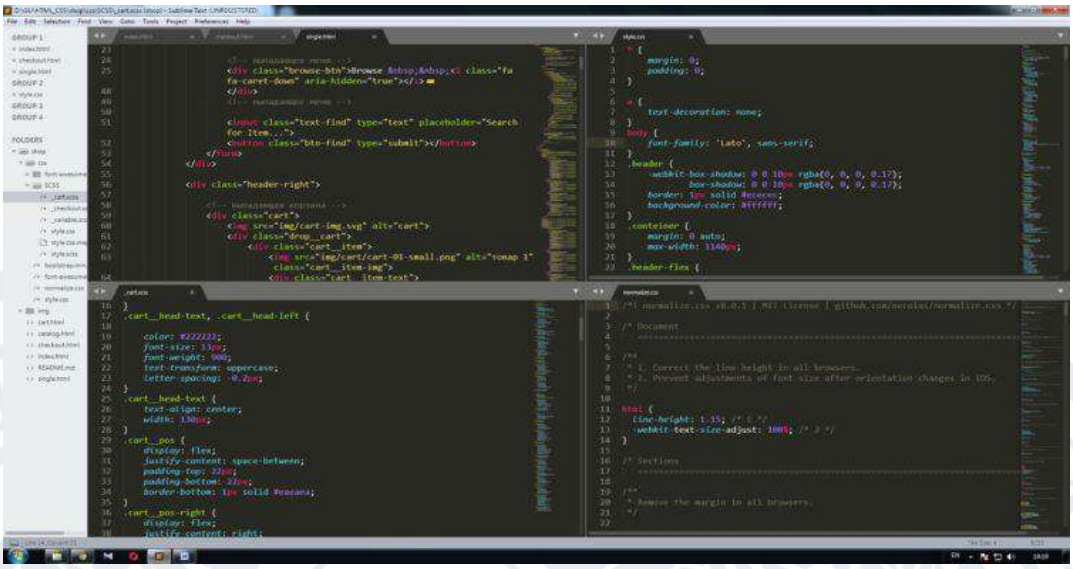


Рисунок 2.2 – Текстовий редактор Sublime Text

WebStorm

WebStorm – це потужне інтегроване середовище розробки від JetBrains, яке спеціалізується на розробці веб-програм та підтримує багато мов програмування, зокрема JavaScript, CoffeScript, TypeScript, і Dart. Воно базується на платформі IntelliJ IDEA і є розширеною версією PhpStorm з підтримкою основних JavaScript розширень.

Основні переваги WebStorm включають в себе:

1. Підтримку інструментів для збірок, таких як Gulp та NPM, що полегшує автоматизацію робочих процесів.

2. Інтеграцію з системою контролю версій GIT, що сприяє ефективній роботі з кодом у команді.

3. Повністю налаштовану робочу область, яка дозволяє користувачам писати код у своєму вигляді.

4. Стартовий пакет із шаблонами для розробки.

5. Розумний редактор, здатний проводити рефакторинг та генерувати код, а також аналізувати його під час роботи.

6. Підтримку модульного тестування та можливість запуску "Run & Debug" для відлагодження коду.

7. Зручний термінал для роботи з контролем версій.

Проте, слід враховувати, що ця програма має високу ціну, що може бути обмежуючим фактором для деяких користувачів. Однак, якщо ваш проект потребує продуктивного та розширеного середовища розробки, WebStorm може бути відмінним вибором, особливо для серйозних професіоналів і компаній, що займаються розробкою програмного забезпечення.

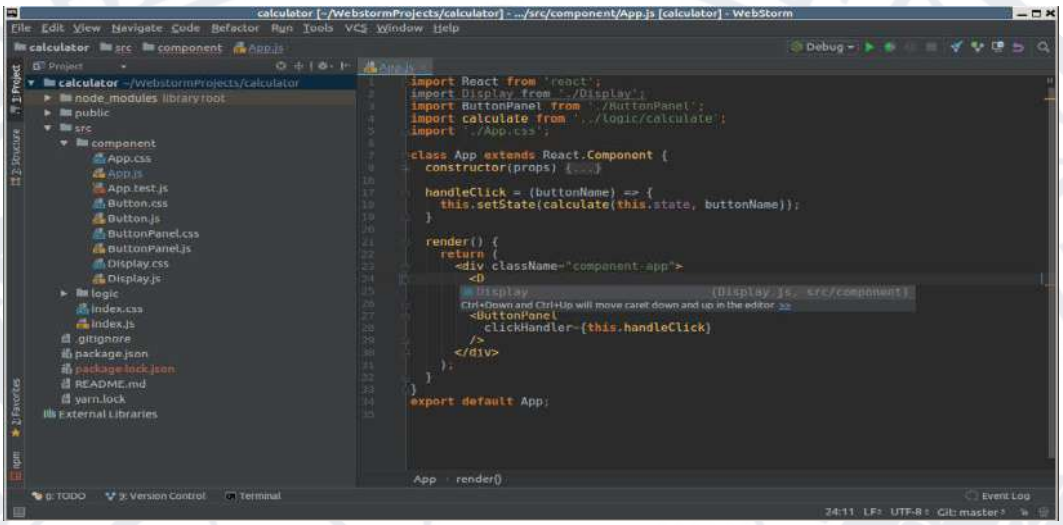


Рисунок 2.3 – проект у WebStorm

Visual Studio Code

Visual Studio Code (VS Code) - це безкоштовне інтегроване середовище розробки, призначене для створення веб-застосунків та програм для хмарних систем. Воно доступне для всіх основних платформ, включаючи Windows, Linux і macOS. Вперше VS Code було представлено на конференції Build 2015 у квітні 2015 року. Основою VS Code є робота над проектом "Atom", який належить компанії GitHub. Ця IDE також ґрунтується на технологіях Atom Shell, яка використовує браузерний двигун Chromium та Node.js.

Основні характеристики VS Code включають в себе:

1. Підтримку різних мов програмування, включаючи JavaScript, TypeScript, Python, Java, і багато інших.

2. Розширені можливості налаштування і розширення завдяки великій кількості плагінів та розширень, що роблять робочий процес більш зручним

3. Інтеграцію з системами контролю версій, такими як Git, для керування кодом та спільної роботи над проектами.

4. Вбудовані інструменти для відлагодження коду (debugging) та роботи з терміналом.

5. Зручний інтерфейс користувача та можливість персоналізації робочого простору.

6. Підтримку розробки веб-застосунків та розширені можливості роботи з хмарними системами.

Основними перевагами VS Code є його безкоштовність та широкий спектр підтримуваних мов програмування. Він став вкрай популярним серед розробників завдяки своїй продуктивності та розширюваності.

Рисунок 2.4 – Логотип Visual Studio Code

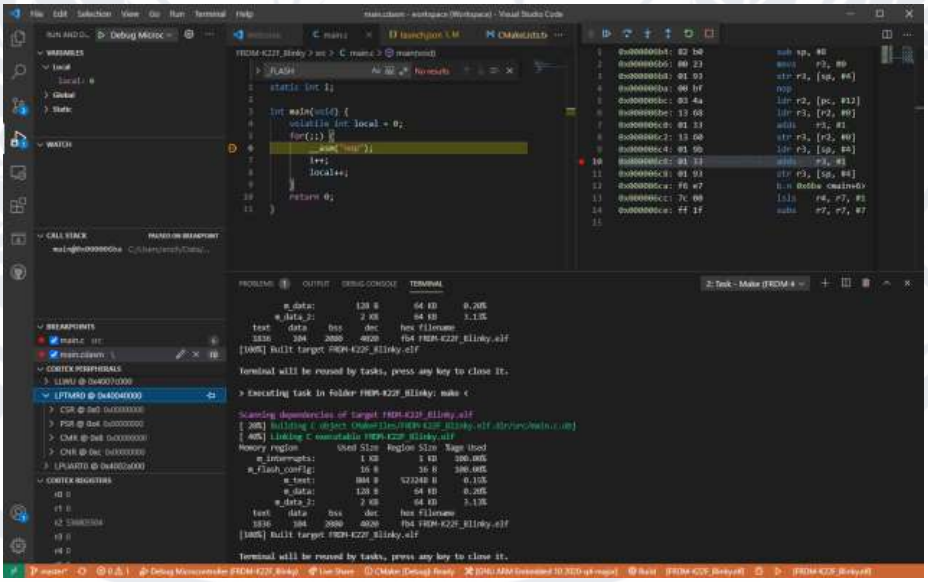


Рисунок 2.5 – Інтерфейс розробки Visual Studio Code

Звісно, Visual Studio Code є дуже зручним, ефективним та потужним інструментом для створення веб-застосунків, і його безкоштовність робить його привабливим вибором для багатьох розробників. Ця IDE надійно визначилася серед інших інструментів завдяки своїй продуктивності та розширюваності.

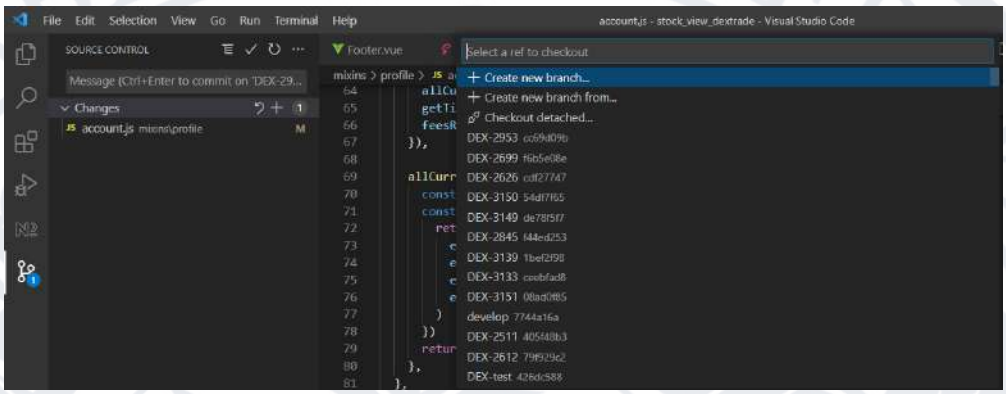


Рисунок 2.6 – робота із Git в VS code

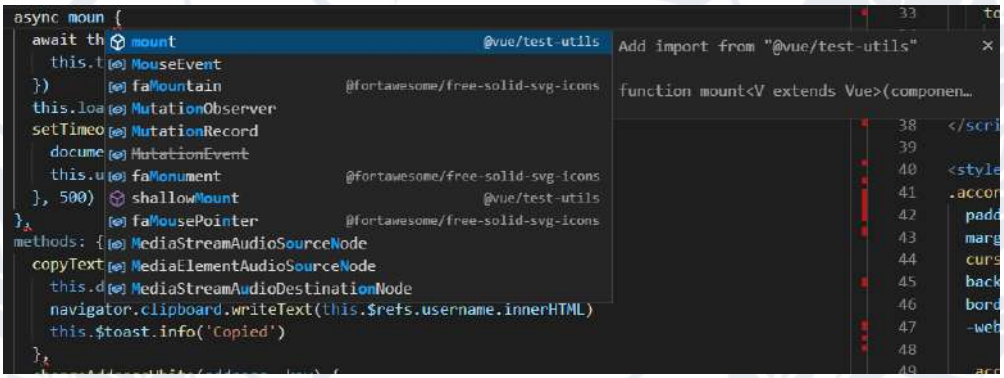


Рисунок 2.7 – Автозавершення відповідно до коду із бібліотек та framework`s

2.2 Технології

Нещодавно веб-програмування було досить доступним для розробників і використовувало базові технології, такі як HTML і CSS. Проте, час йде вперед, і зараз для створення високоякісного продукту потрібно мати значно більше знань. Основною мовою програмування для веб-розробки є JavaScript, але через його обмеження виникає попит на JavaScript-фреймворки, такі як React.js, Angular.js і Vue.js [5].

JavaScript-фреймворк - це інструмент, який можна використовувати для створення сучасних веб-додатків. Раніше, коли інтерфейс додатка був простішим, більшість розробників без труднощів писали його на звичайному JavaScript. Проте з часом інтерфейси та їх логіка стали занадто складними для стандартного JavaScript, і саме тоді почали використовувати фреймворки.

Давайте розглянемо ключові переваги JavaScript-фреймворків:

Веб-компоненти: У веб-розробці використовується компонентний підхід, який є практично стандартом в усіх мовах програмування, за винятком JavaScript. Цей підхід перетворює веб-сайти на повноцінні додатки. Замість необхідності кожен раз повторювати код, наприклад, для створення заголовку сайту, достатньо створити компонент для цього заголовку і підключати його на всіх сторінках лише за допомогою одного рядка коду. Таким чином, ми можемо перевикористовувати будь-яку частину сайту.

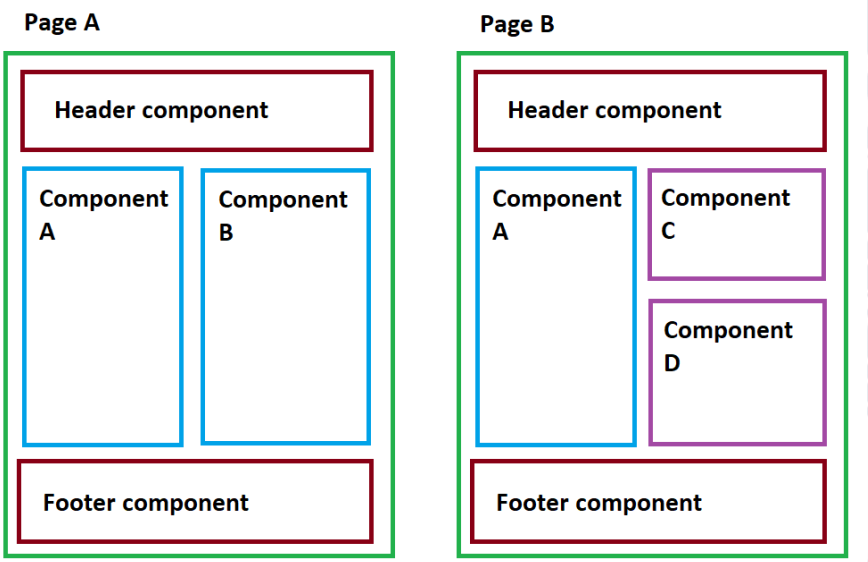


Рисунок 2.8 – Приклад компонентів на Vue.js

Маршрутизація - сучасні фреймворки надають змогу використовувати систему навігації, яка дозволяє переміщатися по веб-сайту без перезавантаження сторінок. Це означає, що ви можете створити багатосторінковий веб-сайт, який при цьому буде мати всі переваги односторінкового застосунку (SPA - Single Page Application).

```
<router-link :to="'/home'">Home</router-link>

<!-- same as above -->
<router-link :to="{ path: '/home' }">Home</router-link>

<!-- named route -->
<router-link :to="{ name: 'user', params: { userId: '123' }}">User</router-link>

<!-- with query, resulting in `/register?plan=private` -->
<router-link :to="{ path: '/register', query: { plan: 'private' }}">
  Register
</router-link>
```

Рисунок 2.9 – Приклад маршрутизації на фреймворках

Зручні інструменти для збірки - компонентний підхід вимагає наявності інструменту, який може об'єднувати всі ці файли в один, а також компілювати їх в розмітку, яку може розуміти браузер. Основні фреймворки мають свої інструменти для збірки, які легко встановлюються і розширюють певний функціонал.

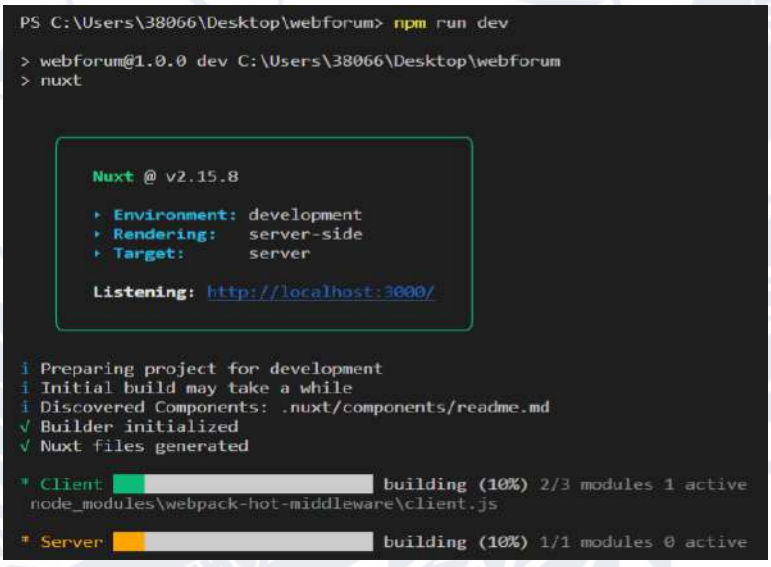


Рисунок 2.10 – Приклад процесу збірки компонентів

Зараз розберемо основні відмінності один від одного, та оберемо фреймворк , на якому будемо розробляти наш веб-додаток для аналізу.

React.js - це бібліотека для створення користувацьких інтерфейсів, і вона не є повноцінним фреймворком. React використовується для створення візуального представлення та взаємодії з іншими бібліотеками. Для створення веб-сторінок зазвичай використовується разом з бібліотекою ReactDOM.

Однією з основних переваг React є можливість розробки великих веб-застосунків, які працюють з даними, що змінюються з часом, без необхідності перезавантаження сторінки. Головна мета React - це забезпечити швидкість, масштабованість і простоту розробки веб-інтерфейсів.

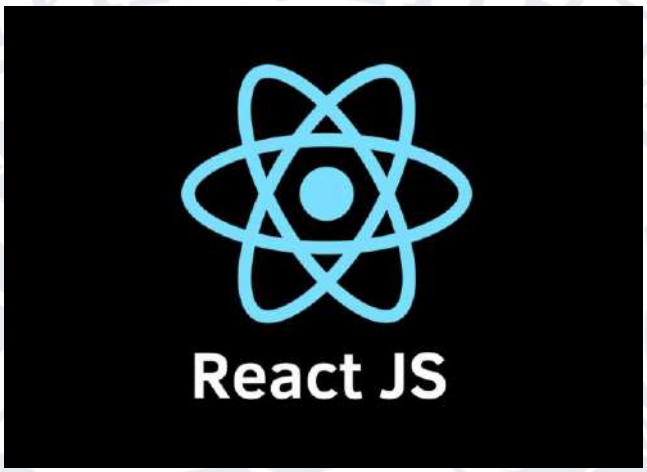


Рисунок 2.11 – Логотип React.js

Переваги React.js

Однією з ключових переваг React.js є використання Віртуальної Об'єктної Моделі Документа (Віртуальної DOM). DOM - це модель документа, яка визначає структуру HTML-документа у формі дерева, і цей документ відправляється сервером клієнту після відповідного запиту. DOM представляє веб-сторінку у вигляді об'єктів, щоб мови програмування могли взаємодіяти з нею. React використовує Віртуальну Об'єктну Модель Документа, яка є внутрішньою репрезентацією DOM у пам'яті. Це дозволяє React працювати набагато ефективніше та швидше. Замість безпосередньої маніпуляції реальним DOM, React оновлює лише ту частину Віртуальної DOM, яка дійсно змінилася, і потім виконує оптимізоване оновлення реального DOM. Це сприяє зменшенню навантаження на браузер і підвищенню продуктивності веб-застосунків, особливо в великих проектах.

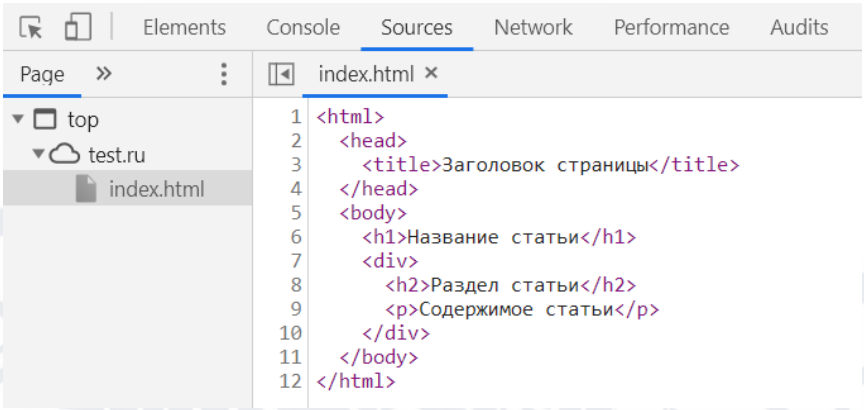


Рисунок 2.12 – Приклад DOM

Браузер регулярно моніторить будь-які зміни в DOM, оновлюючи його відповідно. Зміни в Об'єктній Моделі Документа виникають через різні фактори, такі як користувацький ввід даних, HTTP-запити, отримання даних з API та інші події. Браузер оновлює DOM кожного разу, коли сторінка змінюється. Оскільки сучасні веб-сайти мають великий обсяг DOM, оновлення може займати багато часу і сповільнювати продуктивність веб-додатка. React.js вирішує цю проблему, взаємодіючи зі спрощеною копією DOM, відомою як Віртуальний DOM. Таким чином, реальний DOM оновлюється лише після взаємодії з Віртуальним DOM, що робить оновлення більш ефективним та швидким. Ще однією перевагою React.js є можливість повторного використання компонентів.

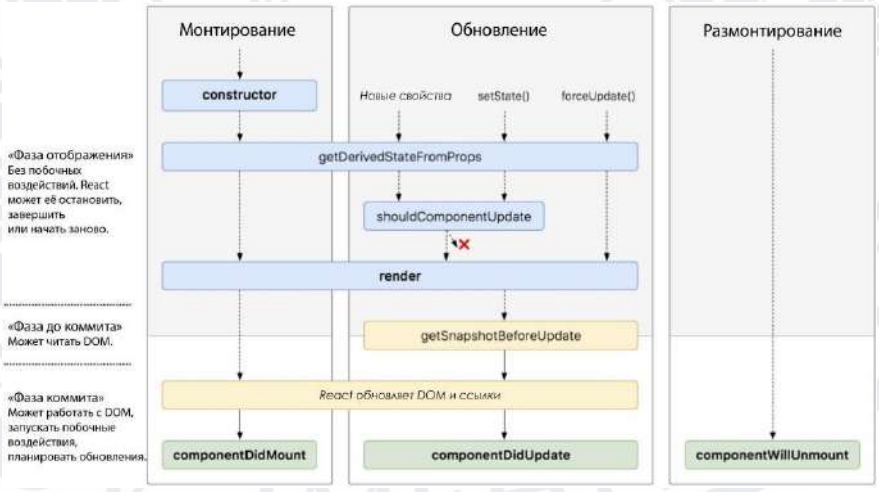


Рисунок 2.13 – Приклад DOM

При роботі з React.js створюються багаторазові компоненти, і часто ці компоненти інтерфейсу користувача можна використовувати в різних частинах коду або навіть в різних проектах практично без змін. Крім того, для розробників React доступні бібліотеки готових компонентів з відкритим вихідним кодом. Завдяки цим бібліотекам, час розробки скорочується, що особливо важливо для стартапів, яким потрібно економити не лише гроші, але і час.

Східний потік даних

Односторонній потік даних в React - це ще одна корисна функція, яку також називають "зверху вниз" або "від батька до дитини". Ця концепція в React дозволяє передавати дані між компонентами лише за одним напрямком. Дані передаються від "батьківського" компонента до "дитячого" компонента, але не навпаки. Такий підхід робить код більш простим та передбачуваним.

Односторонній потік даних також допомагає запобігати помилкам та полегшує налагодження, оскільки він визначає, який компонент відповідає за дані, і якщо щось йде не так, помилку можна легше визначити та виправити.

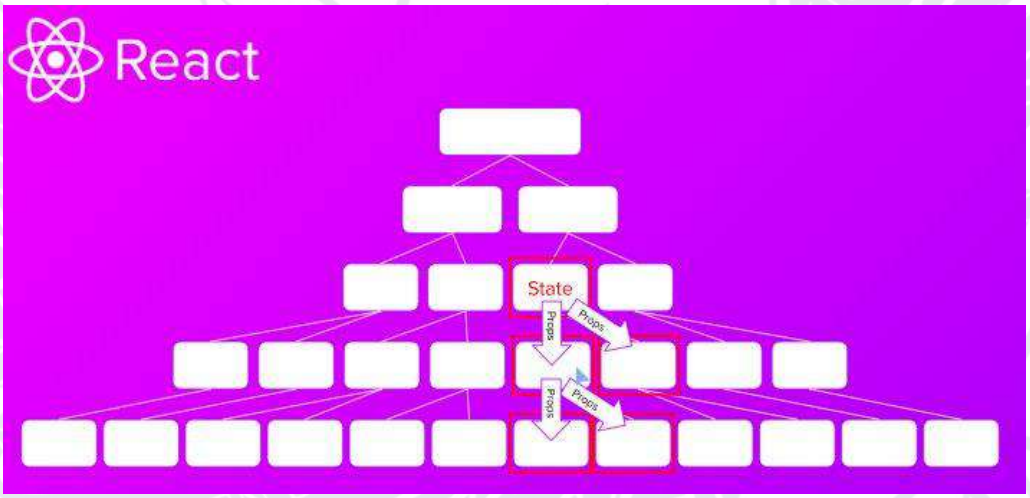


Рисунок 2.14 – Архітектура потоків даних

Браузерні інструменти React-розробника

React Developer Tools – це безкоштовне розширення для Chrome і Firefox , яке презентує цілий набір віджетів перевірки . Розширення спрощує налагодження , дозволяючи розробникам не тільки шукати по списку всіх компонентів , але і переглядати глибоко вкладені компоненти в браузері . [6]

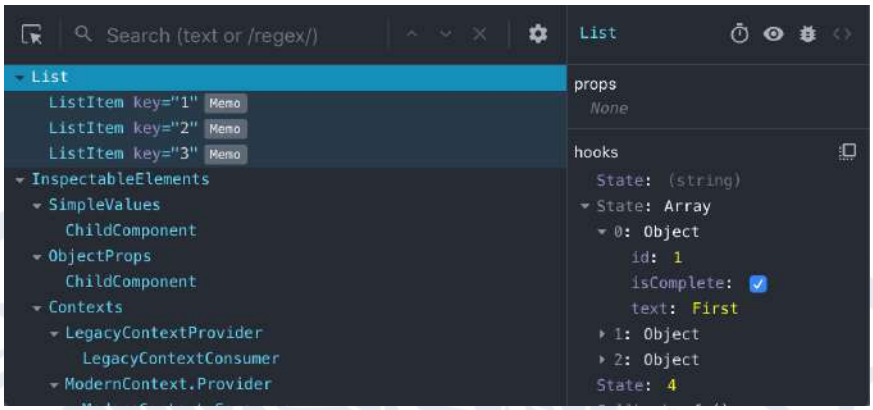


Рисунок 2.15 – React Developer Tools

Недоліки React.js

Як і будь яка інша технологія , React має недоліки :

1 Неповна документація

2 Складний синтаксис JSX

3 Проблеми з пошуковою оптимізацією

4 Фокусування на інтерфейсі користувача

```
class BookList extends Component {

    constructor(props) {
        super(props)
        //this.props = props;
    }

    renderList() {
        return this.props.books.map((book) => {
            return (
                <li key={book.title} className="list-group-item">{book.title}</li>
            )
        })
    }

    render() {
        return (
            <ul className="list-group col-sm-4">
                {this.renderList()}
            </ul>
        )
    }

}

// function is the glue between react and redux
function mapStateToProps(state) {
    // Whatever gets retrieved from here will show up as props inside
    // of book-list

    return {
        books: state.books
    }
}

function mapDispatchToProps(dispatch) {
    return bindActionCreators({selectBook: selectBook}, dispatch)
}

export default connect(mapStateToProps, mapDispatchToProps)(BookList);
```

Рисунок 2.16 – Синтаксис JSX

Як висновок, можна сказати, що дані недоліки є доволі критичні, і його використання не є оптимальним рішенням.

Angular

Написаний на TypeScript фронтенд-фреймворк з відкритим вихідним кодом, який розробляється під керівництвом команди Angular, був заархівований 18 серпня 2021 року і зберігається в Weyback Machine, в компанії Google, а також активно підтримується спільнотою приватних розробників та корпорацій.
До його переваг можна віднести наступне:

Декларативний стиль коду. При створенні шаблонів у фреймворку Angular.js використовується декларативна парадигма програмування. Це означає, що код описує бажаний кінцевий результат, а не всі кроки, які потрібно виконати для його досягнення. Такий підхід робить код більш легким і зрозумілим, полегшує читання та підтримку коду. Замість того, щоб докладно описувати кожну дію, програмісти можуть зосередитися на тому, який результат вони хочуть отримати, що дозволяє зробити код більш чистим і зрозумілим.

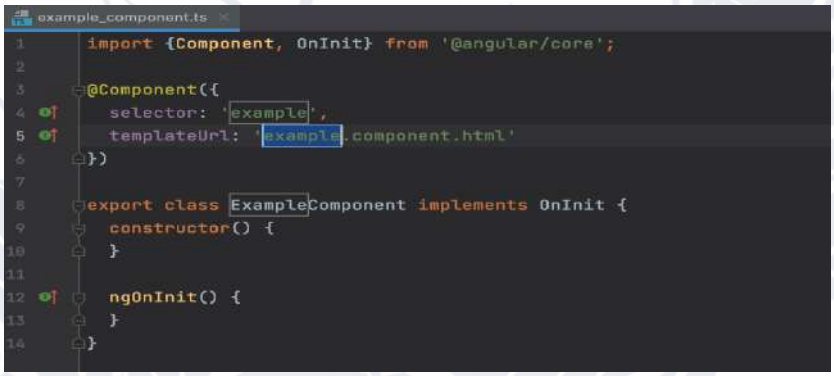


Рисунок 2.17 – Приклад коду на Angular

Використання директив. У фреймворку Angular.js мовою шаблонів є HTML. HTML розширюється за допомогою директив, які додають в код інформацію про необхідну поведінку, наприклад, про потребу в завантаженні певного модуля одразу після завантаження сторінки. Директиви дозволяють розробникам зосередитися на обробці логіки та працювати більш продуктивно. Їх можна використовувати повторно, що також полегшує читання коду і сприяє його читабельності.

Рисунок 2.18 – Директиви на Angular.js

Висока швидкість розробки. Правильно використовуючи Angular.js, можна швидко розробляти великі програми. Фреймворк Angular.js надає інструменти і підходи, що сприяють організації коду, зменшують дублювання та дозволяють ефективно взаємодіяти з сервером та іншими компонентами програми. Такий підхід полегшує розробку та підтримку великих програм, а також допомагає зберігати код організованим і легким для розуміння.

MVC з коробки. В AngularJS використовується схема MVC (Model-View-Controller), яка допомагає розділити логіку, представлення та дані програми на окремі компоненти. [7] Схема Модель-Вид-Контролер (MVC) в Angular.js дозволяє створювати односторінкові веб-програми (Single Page Applications, SPA). Фреймворк Angular.js надає службу $http, яка спрощує взаємодію з віддаленими HTTP-серверами за допомогою XMLHttpRequest або JSONP. При передачі об'єкта JavaScript на сервер, він автоматично перетворюється у рядок JSON, і після отримання відповіді служба також спробує перетворити отриманий рядок JSON назад у JavaScript об'єкт. За допомогою служби $http, ви можете створити власні служби з повним контролем над обробкою URL та даних для взаємодії з сервером.

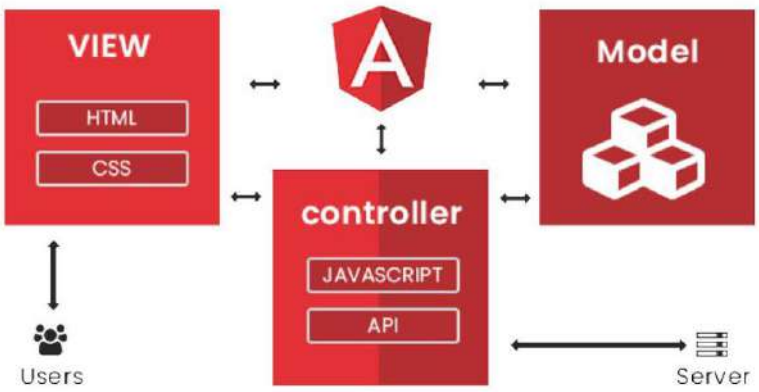


Рисунок 2.19 – Структура для SPA

Справді, AngularJS, як і будь-який інший фреймворк або бібліотека, має свої недоліки:

1. Складність освоєння: AngularJS може виявитися складним для освоєння, особливо для тих, хто звик працювати з бібліотекою jQuery або іншими інструментами, які використовують маніпуляції з DOM-деревом. Він вимагає від розробника засвоєння нового підходу до структури та організації коду.

2. Уповільнення роботи при великій кількості вотчерів: При використанні багатьох вотчерів (або слухачів подій) в програмі може виникнути уповільнення продуктивності, особливо на стороні клієнта. Це може призвести до збільшення часу реакції програми.

3. Відсутність зворотної сумісності з другою версію: Однією з проблем AngularJS була відсутність зворотної сумісності з Angular 2 і пізнішими версіями. Це означало, що проекти, які використовують AngularJS, мали б пройти процес міграції до нових версій, що може бути складним і вимагати часу та зусиль.

Не дивлячись на ці недоліки, AngularJS також має свої переваги та застосування, і для багатьох проектів він може бути досить ефективним інструментом розробки.

Vue.js

Vue.js - це ще один популярний JavaScript фреймворк для створення користувацьких інтерфейсів. Він має ядро, яке включає в себе бібліотеку ядра та

маршрутизатор. Один з основних плюсів Vue.js - це можливість інтегрувати його поступово в вашу систему, що відрізняє його від інших монолітичних аналогів.

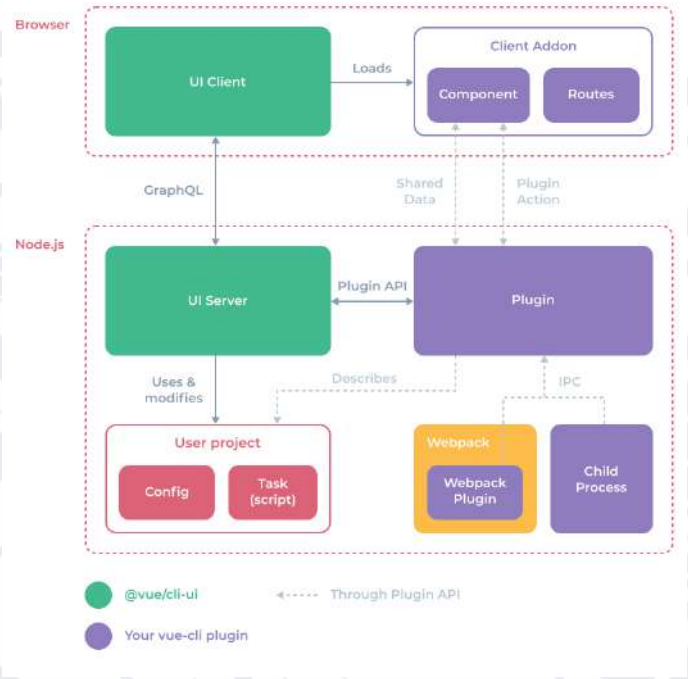


Рисунок 2.20 – Структура Vue.js

Директиви в Vue.js - це спеціальні атрибути, які використовуються для додавання логіки до елементів HTML. У Vue.js директиви розпізнаються за префіксом "v-", і вони можуть мати аргументи, які представляють собою атрибути HTML або події. Директиви дозволяють зв'язувати дані з DOM, керувати відображенням елементів, виконувати анімації та багато інших речей, що полегшують роботу з інтерфейсом користувача.

Розглянемо основні з них.

V-bind – динамічно зв'язується з одним або декількома атрибутами.

V-cloak - ховає "вусаті" вирази, поки не підтягнулися дані

v-if – умова для рендеру елемента

V-else - означає "else блок" для v-if

V-for - циклічно проходить масив об'єктів

V-model – зв'язує стан з input елементом

V-on – пов'язує слухача події з елементом

V-pre – не компілює елемент та його дочірні елементи

V-show – перемикає видимість елемента, змінюючи властивість CSS display

V-text — оновлює textContent елемент

```
<div id="app">
  <div v-if="flag">foo</div>
  <div v-else>bar</div>
  <button @click="flag=!flag">Click Me!</button>
</div>
<script type="text/javascript">
new Vue({
  el: '#app',
  data:{
    flag : true
  }
})
</script>
```

Рисунок 2.21 – Приклад директивів Vue.js

Так, компоненти є однією з ключових зручностей багатьох фреймворків, включаючи Vue.js. Компоненти дозволяють розділити веб-інтерфейс на малий, самодостатній та перевикористовуваний кодові блоки. Це дозволяє полегшити розробку та підтримку великих додатків, оскільки ви можете працювати з окремими частинами інтерфейсу незалежно од одних від інших.

Компоненти в Vue.js можуть бути використані для розширення стандартних HTML-елементів або створення власних користувацьких елементів інтерфейсу. Вони дозволяють вам легко створювати та використовувати багаторазові UI-елементи, такі як кнопки, форми, таблиці і т. д.

Це допомагає полегшити процес розробки, підтримки та рефакторингу коду, особливо великих проектів, і сприяє покращенню структури та організації коду..

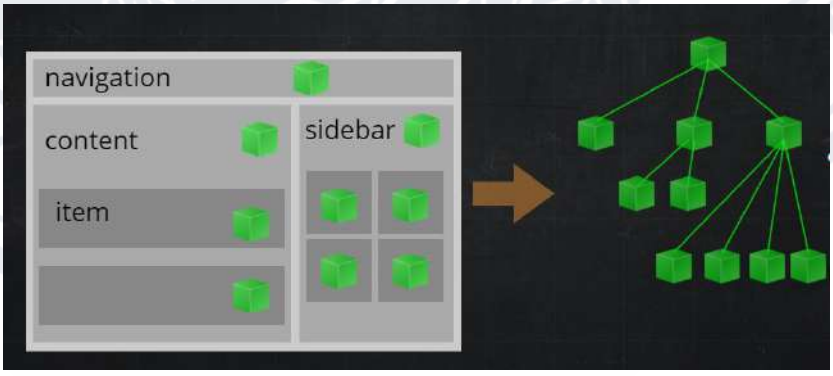


Рисунок 2.22 – Приклад структури компонентного підходу Vue.js

Переходи

Vue.js дійсно надає широкий спектр інструментів для створення анімацій та покращення візуальної привабливості веб-додатків. Завдяки цим можливостям розробники можуть легко додавати анімації до різних елементів сторінки та зробити користувацький інтерфейс більш інтерактивним і привабливим для користувачів. Важливою перевагою Vue.js є можливість автоматичного застосування класів для CSS-переходів і анімацій. Це дозволяє вам легко визначати анімаційні ефекти, використовуючи CSS, і мати контроль над ними без необхідності писати багато JavaScript-коду. Крім того, можливість інтеграції сторонніх бібліотек для CSS-анімацій, таких як "Animate.css", та JavaScript-бібліотек для анімацій, таких як "Velocity.js", розширює можливості створення різноманітних ефектів. Загалом, ці інструменти дозволяють розробникам створювати власні, унікальні анімації і забезпечувати привабливий та інтерактивний дизайн для веб-додатків.

Рисунок 2.23 – Використання анімацій у Vue.js

Роутинг

Так, Vue.js має спеціальний пакет Vue Router, який дозволяє розробникам керувати маршрутизацією та навігацією в їх веб-додатках. Vue Router надає зручний і спрощений спосіб налаштування маршрутів та інтеграцію вкладених маршрутів до вкладених компонентів.

Основні функціональність Vue Router включають:

1. Визначення маршрутів: Розробники можуть визначити маршрути та відповідні компоненти для кожного маршруту. Наприклад, вони можуть налаштовувати маршрути для різних сторінок вашого додатка.

2. Вкладені маршрути: Vue Router підтримує вкладені маршрути, що дозволяє вбудовувати компоненти в інші компоненти та створювати складні ієрархії сторінок.

3. Хуки навігації: Ви можете використовувати хуки навігації, такі як `beforeEach` та `beforeResolve`, для виконання дій перед навігацією до певної сторінки.

4. Параметри маршруту: Ви можете передавати та отримувати параметри у URL-адресах маршрутів, щоб динамічно змінювати контент сторінки.

5. Керування історією браузера: Vue Router дозволяє вам керувати історією браузера, використовуючи режими "hash" або "history", що дозволяє створювати плавну навігацію без перезавантаження сторінок.

Усі ці можливості роблять Vue Router потужним і зручним інструментом для управління маршрутизацією в вашому Vue.js додатку. [8]

app.js

```
import Vue from 'vue'
import VueRouter from 'vue-router'
import App from './app.vue'
import ViewA from './view-a.vue'
import ViewB from './view-b.vue'

Vue.use(VueRouter)

const router = new VueRouter()

router.map({
  '/a': { component: ViewA },
  '/b': { component: ViewB }
})

router.start(App, '#app')
```

Рисунок 2.24 – Приклад використання маршрутизації.

Ajax – запити

```
{
  // GET /someUrl
  this.$http.get('/someUrl').then((response) => {
    // успех
  }, (response) => {
    // или ошибка
  });
}
```

Рисунок 2.25 – Використання vue-resource

Управління станом через Vuex

Vuex - це потужна бібліотека та паттерн управління станом для додатків на Vue.js. В основі Vuex лежить централізований стан, який використовується для зберігання даних, доступних для всіх компонентів у додатку. Це дозволяє створювати передбачувані та консистентні зміни стану додатка.

Структура Vuex включає наступні ключові елементи:

1. Стан (State): Це єдине джерело даних для всіх компонентів. Стан представляє собою об'єкт, що містить всі дані, які можуть бути використані у додатку.

2. Компоненти (Components): Компоненти є декларативними відображеннями стану. Вони отримують доступ до стану через спеціальні властивості та можуть відображати дані із стану у користувацькому інтерфейсі.

3. Дії (Actions): Дії виконуються у відповідь на події, які виникають у додатку. Вони можуть збирати дані з інших джерел, наприклад, із зовнішніх API, та потім викликати мутації для зміни стану.

4. Мутації (Mutations): Мутації - це функції, які змінюють стан. Вони приймають поточний стан та дані, щоб виконати зміни у стані. Мутації служать для забезпечення консистентних та передбачуваних змін стану.

Vuex робить управління станом додатка більш простим та ефективним, особливо для великих та складних додатків. Він допомагає уникнути проблем, пов'язаних із зміною стану і спрощує відлагодження та підтримку коду.

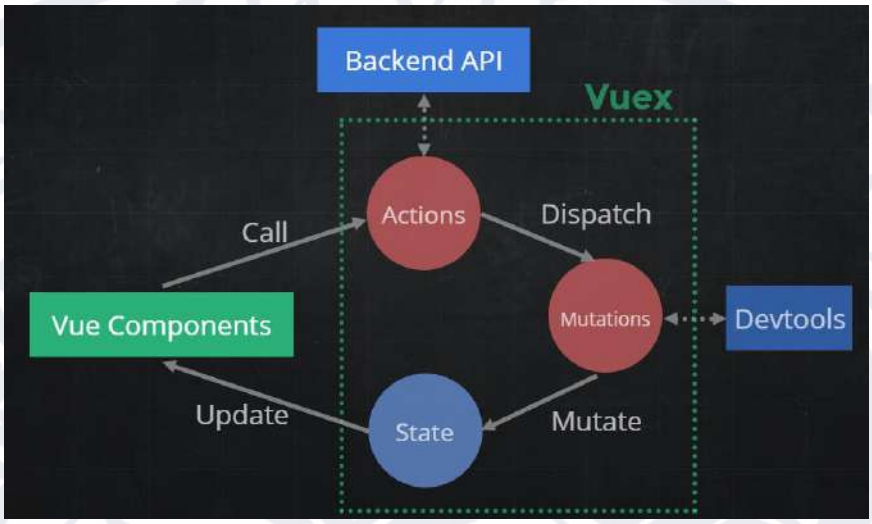


Рисунок 2.26 – Приклад роботи структури Vuex

Висновок полягає в тому, що обраний фреймворк є найбільш прогресивним і потужним інструментом для розробки вашого додатку. Однак важливо враховувати, що фреймворк - це лише одна з технологій, яку можна використовувати у розробці. Для створення повноцінного веб-сайту, ймовірно, потрібно буде комбінувати фреймворк з іншими технологіями і інструментами, які найкраще підходять до вашого проекту. Це може включати в себе різні бібліотеки, засоби для роботи з базами даних, інструменти для створення користувацького інтерфейсу та багато інших компонентів.

HTML 5

HTML5 є сучасною мовою розмітки гіпертексту, і вона є стандартом для розробки веб-сторінок. Вона виступає як фундамент для будь-якого веб-сайту і важлива для забезпечення якості та ефективності вашого форуму. Під час розробки форуму дотримання найкращих практик використання HTML5 може забезпечити такі переваги, як:

1. Валідність: Використання правильної HTML5 розмітки допомагає уникнути помилок і забезпечує, що ваша сторінка буде сумісною з різними браузерами.

2. Семантика: HTML5 має великий набір семантичних елементів, таких як `<header>`, `<nav>`, `<article>`, `<section>` і багато інших. Використання цих елементів допомагає визначити структуру сторінки та полегшити її розуміння як браузерами, так і розробниками.

3. Доступність: Правильне використання HTML5 може покращити доступність вашого форуму для користувачів з обмеженими можливостями, такими як використання семантичних елементів та атрибутів `aria`.

4. Мультимедіа: HTML5 також приніс підтримку для відео та аудіо без сторонніх плагінів, що полегшує відтворення мультимедійних контентів на веб-сторінках.

5. Локальне сховище: HTML5 пропонує можливості локального сховища, такі як `localStorage` і `sessionStorage`, що дозволяють зберігати дані клієнта на стороні клієнта.

Загалом, використання HTML5 разом з іншими сучасними технологіями і фреймворками допоможе створити додаток, який буде не лише функціональним і ефективним, але й сучасним і сумісним зі стандартами розробки веб-сайтів.

Рисунок 2.27 – Логотип HTMЛ

CSS3

(Каскадні таблиці стилів) є найновішою версією мови CSS і використовується для задання стилістичних особливостей веб-сторінок і документів. Вона пропонує багато покращень порівняно з попередніми версіями CSS, включаючи нові можливості для створення більш складних та креативних дизайнів. Основні особливості CSS3 включають:

1. Візуальні ефекти: CSS3 дозволяє створювати візуальні ефекти, такі як тіні, градієнти, анімація, перетини і трансформації, без необхідності використання зображень або скриптів. Це дозволяє створювати більш динамічні та привабливі веб-сайти.

2. Респонсивний дизайн: CSS3 має можливості для створення респонсивного дизайну, що дозволяє сторінкам адаптуватися до різних розмірів екрану і пристроїв. Це особливо важливо в епоху мобільних пристроїв.

3. Трансформації і анімації: CSS3 дозволяє обертати, масштабувати, нахиляти і переміщувати елементи сторінки, а також створювати складні анімації за допомогою ключових кадрів (keyframes).

4. Гнучкі шрифти: CSS3 дозволяє використовувати веб-шрифти, що дозволяє розширювати набір шрифтів, які можна використовувати на веб-сайтах.

5. Медіа-запити: CSS3 включає медіа-запити, які дозволяють стилізувати сторінку в залежності від параметрів пристрою, таких як розмір екрану і орієнтація.

6. Гнучкий бокс-модель: CSS3 пропонує більше гнучкості в роботі з бокс-моделлю, включаючи можливість задавати розміри, поля і внутрішні відступи для елементів більш точно і зручно.

CSS3 є важливою частиною сучасної веб-розробки і дозволяє розробникам створювати більш креативні та ефективні стилі для своїх веб-сторінок.

Рисунок 2.28 – Логотип CSS 3

JavaScript є однією з найпоширеніших мов програмування, і вона використовується для взаємодії з користувачем на веб-сторінках. JavaScript дозволяє реалізувати різноманітні функції на веб-сайтах, такі як валідація форм, анімація, маніпуляція DOM, взаємодія з віддаленими серверами (через AJAX запити), створення інтерактивних елементів і багато іншого.

JavaScript використовується як на клієнтській стороні (браузер) так і на серверній стороні (Node.js) для розробки різних видів додатків. Вона є мовою програмування з високим рівнем гнучкості і може бути використана для створення веб-сайтів, веб-додатків, мобільних додатків, інтернет-ігор, і багатьох інших застосувань.

JavaScript також використовується разом із фреймворками і бібліотеками, такими як React, Vue.js, Angular, для полегшення розробки веб-додатків і вдосконалення їхніх можливостей. Вона є невід'ємною частиною сучасного веб-розробництва і важливим інструментом для створення веб-застосунків, які надають користувачам багатий інтерактивний досвід.

Рисунок 2.29 – Логотип JavaScript

Figma - це потужний інструмент для дизайну та прототипування, який використовується для створення макетів сторінок та інтерфейсів веб-сайтів і додатків. Він пропонує розширені можливості для роботи над дизайном та спільної роботи над проектами в реальному часі. Основні переваги Figma включають:

1. Онлайн доступ: Figma доступний як онлайн-сервіс, що означає, що ви можете працювати з ним у будь-якому браузері, без необхідності завантаження та встановлення додатків.

2. Спільна робота: Figma дозволяє різним користувачам працювати над проектом одночасно, роблячи зміни в реальному часі. Це робить його ідеальним інструментом для командної роботи.

3. Прототипування: Ви можете створювати інтерактивні прототипи для відображення функціональності вашого веб-сайту або додатка. Це допомагає вам тестувати ідеї та зміни в інтерфейсі перед реалізацією.

4. Зручність використання: Figma надає інтуїтивний інтерфейс і різноманітні інструменти для роботи з векторними графіками, текстом, кольорами і іншими дизайн-елементами.

5. Зберігання в хмарі: Ваші проекти зберігаються в хмарі, що робить їх доступними з будь-якого пристрою та забезпечує зручний доступ до робочих файлів.

Figma є важливим інструментом для дизайнерів і розробників веб-сайтів, оскільки він дозволяє створювати високоякісні дизайни та прототипи, спільно працювати над проектами та швидко переносити їх в життя.

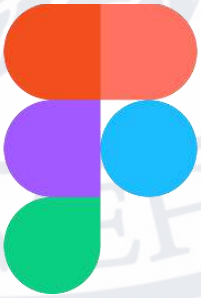


Рисунок 2.30 – Логотип «Figma»

2.3 Оптимізація швидкості завантаження

Оптимізація швидкості завантаження веб-сайту є важливим етапом покращення його функціональності. Швидкість завантаження впливає на користувацький досвід, конверсію, позиції в пошукових системах та загальну продуктивність веб-сайту.

Зменшення розміру файлів: Великі розміри файлів, такі як зображення, JavaScript- та CSS-файли, можуть значно уповільнювати завантаження сторінок. Щоб зменшити розмір файлів, можна використовувати методи стиснення, такі як gzip або Brotli для стиснення текстових файлів, а також оптимізацію та стиснення зображень без втрати якості.

Кешування ресурсів: Використання кешування дозволяє зберігати копії ресурсів (наприклад, зображень, CSS- та JavaScript-файлів) на бічному пристрої користувача. Це дозволяє зменшити кількість запитів до сервера при повторному відвідуванні сторінок, що прискорює завантаження. Для цього можна використовувати HTTP-заголовки, такі як "Expires" або "Cache-Control".

Використання CDN (Content Delivery Network): CDN - це мережа серверів, розташованих у різних географічних областях, які зберігають копії вашого веб-сайту. Використання CDN дозволяє збільшити швидкість завантаження, оскільки веб-сторінки доставляються з сервера, який фізично розташований ближче до користувача. Це особливо корисно для веб-сайтів зі значним трафіком та відвідувачами з різних країн.

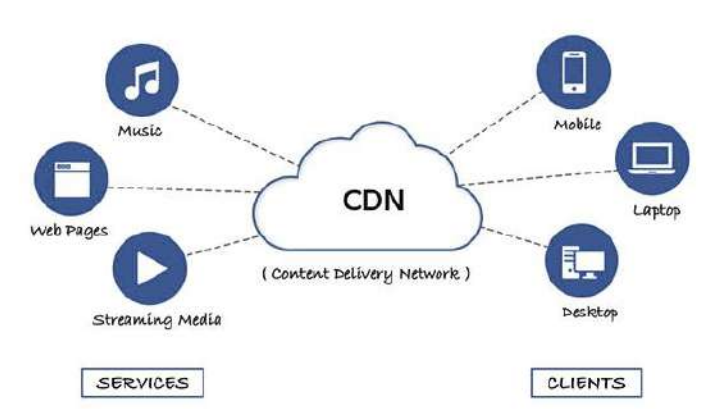


Рисунок 2.31 – Схема роботи CDN

Оптимізація коду: Перевірка та оптимізація коду веб-сторінок може допомогти покращити швидкість завантаження. Це може включати мініфікацію та об'єднання JavaScript- та CSS-файлів, видалення непотрібного коду, якщо він не використовується, а також оптимізацію запитів до бази даних, щоб зменшити затримки при завантаженні сторінок.

Асинхронне завантаження ресурсів: Використання асинхронного завантаження для певних ресурсів, таких як скрипти або зовнішні файли, дозволяє продовжувати завантаження сторінки без очікування завершення завантаження цих ресурсів. Це може покращити загальний час завантаження сторінки та користувацький досвід.

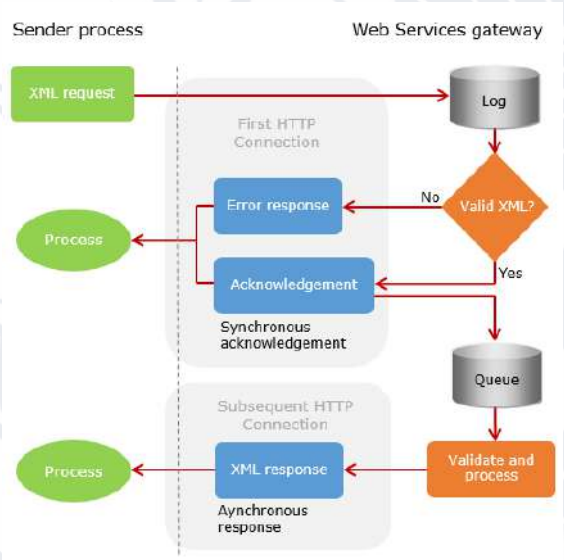


Рисунок 2.32 – Процес роботи з асинхроним кодом

Аналіз швидкості завантаження: Для оцінки ефективності оптимізації швидкості завантаження можна використовувати різні інструменти аналізу швидкості, такі як PageSpeed Insights, Lighthouse або WebPageTest. Вони надають детальну інформацію про швидкість завантаження, рекомендації щодо покращень та показники продуктивності.

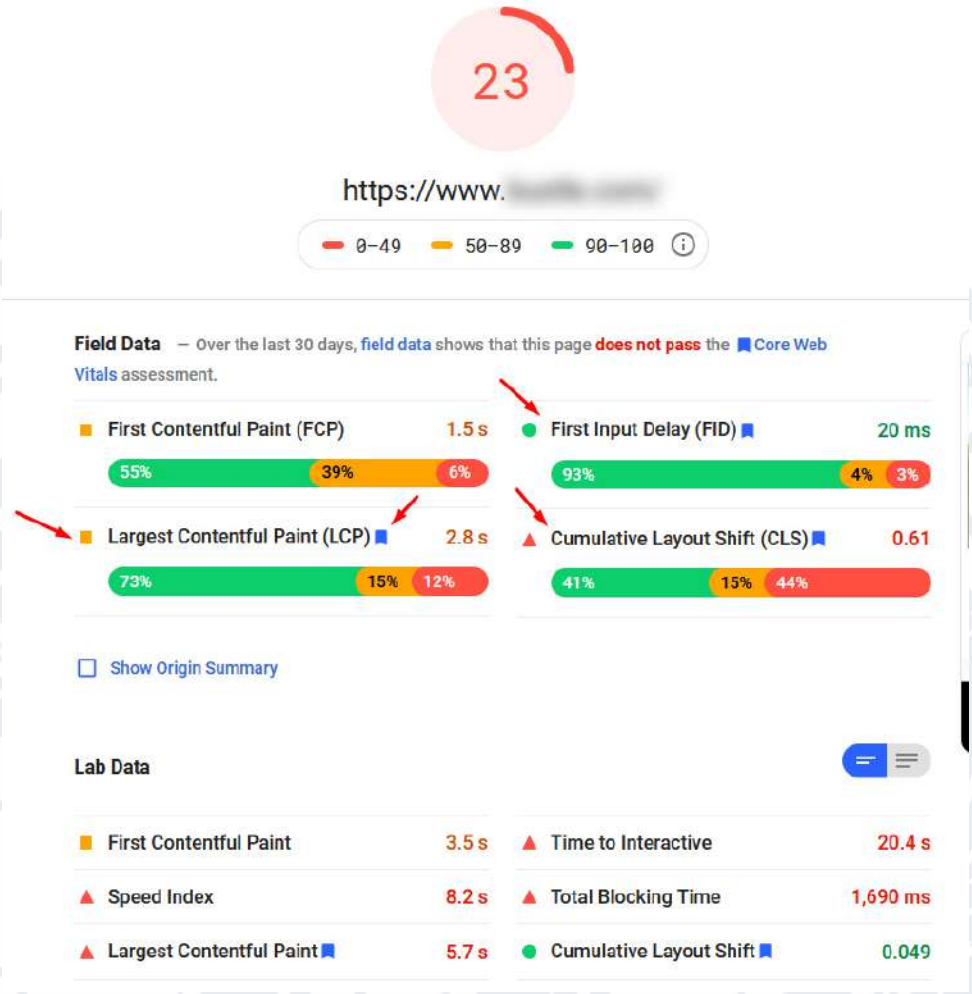


Рисунок 2.33 – Аналіз швидкості

Ці підходи до оптимізації швидкості завантаження веб-сайту можуть бути використані для аналізу та вдосконалення функціональності веб-сайтів.

2.4 Впровадження персоналізації

Впровадження персоналізації на веб-сайті є одним з ключових методів покращення його функціональності. Персоналізація полягає в наданні унікального та індивідуального досвіду користувачам, враховуючи їхній контекст, потреби, поведінку та вподобання. Ось деякі аспекти та методи впровадження персоналізації:

Збір та аналіз даних: Для ефективної персоналізації необхідно збирати та аналізувати дані про користувачів. Це можуть бути дані про їхню реєстрацію, відвідування, покупки, взаємодію з контентом та інші контекстуальні дані.

Застосування аналітики дозволить отримати уявлення про потреби та інтереси користувачів.

Сегментація аудиторії: На основі зібраних даних можна провести сегментацію аудиторії, тобто поділити користувачів на групи з подібними характеристиками. Це дозволить вам створити спеціальні сегменти та пропонувати персоналізований контент для кожної групи.

Динамічний контент: Використання динамічного контенту дозволяє на льоту змінювати вміст сторінок в залежності від індивідуальних характеристик користувачів. Це можуть бути рекомендації товарів, персоналізовані пропозиції, повідомлення про новини або події, які цікавлять конкретного користувача.

Рекомендації на основі інтересів: Використання алгоритмів рекомендацій дозволяє пропонувати користувачам контент, який їм може бути цікавим на основі їхніх попередніх взаємодій, покупок або переглядів. Це може бути використано для рекомендацій товарів, статей, відео та інших ресурсів.

Адаптивний дизайн: Використання адаптивного дизайну дозволяє адаптувати вигляд та розміщення елементів веб-сайту в залежності від характеристик користувача, таких як розмір екрану, пристрій або орієнтація. Це забезпечує оптимальний досвід перегляду незалежно від використовуваного пристрою.

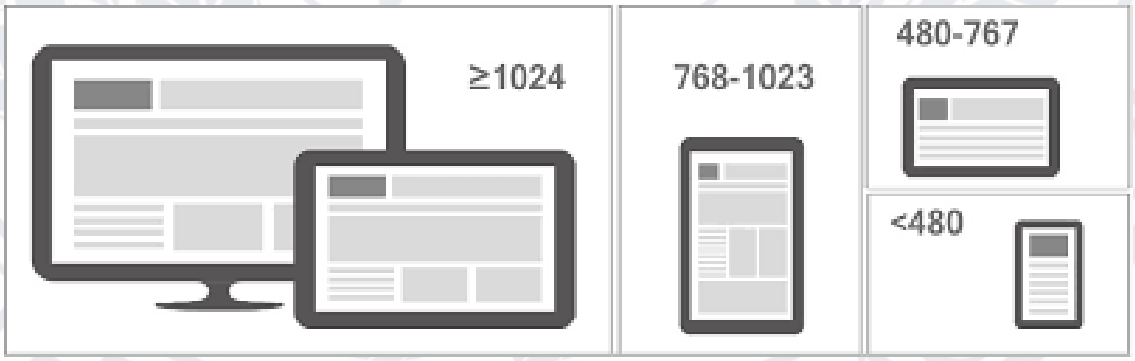


Рисунок 2.34 – Схема розмірів адаптивних дизайнів

A/B-тестування: Використання методу A/B-тестування дозволяє порівняти ефективність різних варіантів персоналізації. Шляхом тестування різних

варіантів ви можете визначити, який підхід працює краще та має більший вплив на користувачів.

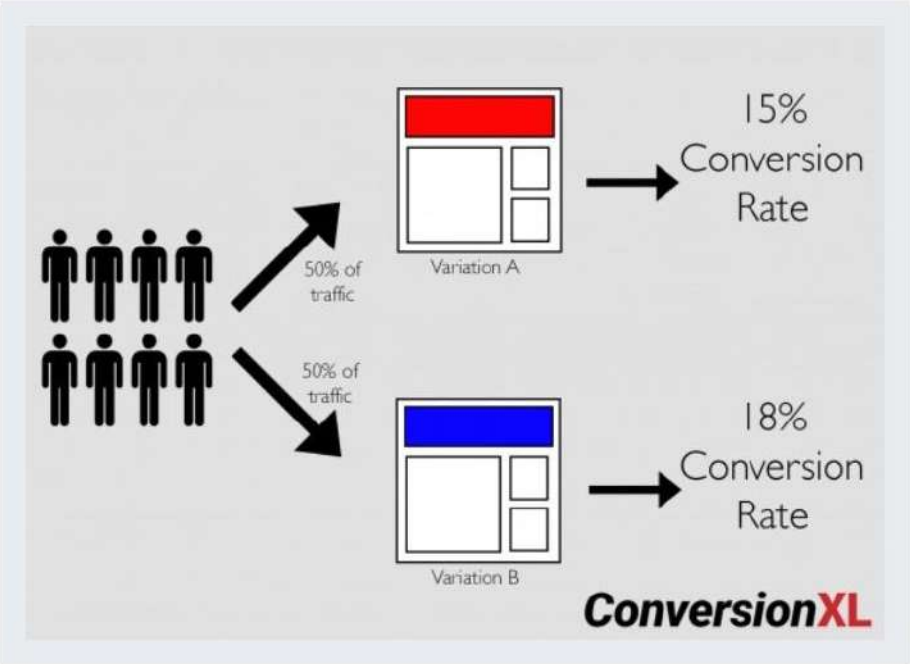


Рисунок 2.35 – Схема A/B тестування

Машинне навчання: Використання методів машинного навчання дозволяє аналізувати дані про користувачів та прогнозувати їхні дії та вподобання. Це може бути використано для покращення персоналізації та автоматичного пропонування відповідного контенту.

Впровадження персоналізації на веб-сайті дозволяє створити більш індивідуальний та привабливий досвід для користувачів. Це може позитивно позначитись на взаємодії користувачів з веб-сайтом, їх задоволенні та конверсії. Використання вищезгаданих методів дозволить вам більш науково та математично підходити до реалізації персоналізації та оцінити її ефективність.

2.5 Використання наукових та математичних методів аналізу та покращення

Мультиваріантне тестування є одним з ключових наукових методів, що дозволяє проводити експерименти з різними варіантами персоналізації та порівнювати їх ефективність. У цьому методі випадковим чином відбираються

групи користувачів, які бачать різні варіанти веб-сайту з різними рівнями персоналізації, а потім аналізуються їхні реакції та показники ефективності. На основі результатів тестування можна зробити об'єктивне рішення про те, яка стратегія персоналізації працює краще.

Аналіз варіацій є ще одним математичним підходом, що дозволяє вивчати вплив різних факторів на ефективність персоналізації. Цей метод дозволяє розбити експеримент на окремі фактори та виміряти їхній вплив на показники, що оцінюють ефективність. Це допомагає зрозуміти, які фактори мають найбільший вплив і на яких елементах веб-сайту слід зосередитись для покращення.

Статистичні методи, такі як аналіз регресії, кореляційний аналіз та статистичне моделювання, можуть допомогти в оцінці статистичної значущості змін та виявленні залежностей між різними змінними. Вони дозволяють провести глибокий аналіз даних та зробити науково обґрунтовані висновки щодо ефективності різних стратегій та методів персоналізації.

Математичні моделі можуть бути використані для прогнозування результатів різних стратегій персоналізації на основі даних та статистики. Вони дозволяють розробити моделі, які описують взаємодію користувачів з веб-сайтом та вплив персоналізації на їхні дії та поведінку. Це може допомогти прогнозувати ефективність різних стратегій та зробити обґрунтовані рішення щодо впровадження персоналізації. Використання наукових та математичних методів у покращенні персоналізації дозволяє підходити до цього процесу більш об'єктивно та обґрунтовано. Вони допомагають зрозуміти причинно-наслідкові зв'язки, виявити найефективніші стратегії та забезпечити оптимальний досвід користувача.

Висновок до другого розділу

Ми розглянули та порівняли всі необхідні інструменти та технології для вирішення нашої задачі. Зрозуміли різницю між тими чи іншими технологіями та обрали найбільш правильні технології для розробки додатка.

РОЗДІЛ 3
АНАЛІЗ ДАНИХ ТА НАДАННЯ РЕКОМЕНДАЦІЙ

3.1 Створення інтерфейсу

Розробка інтерфейсу визначає взаємодію користувача з веб-ресурсом, і правильний дизайн відіграє критичну роль у створенні позитивного враження та комфортної взаємодії з платформою. Принципи UI/UX дизайну визнаються ключовими факторами для створення ефективного та конкурентоспроможного веб-продукту.

Перші враження:

Дизайн інтерфейсу є першим, що бачить користувач під час візиту на веб-ресурс. Перші враження важливі, оскільки вони впливають на відношення користувача до сайту чи додатку. Привабливий та легкий для сприйняття дизайн створює позитивний тон та заохочує користувача досліджувати далі.

Комфортна взаємодія:

Ефективний дизайн забезпечує комфортну взаємодію з ресурсом. Інтуїтивно зрозумілі елементи інтерфейсу дозволяють користувачам швидко зорієнтуватися та використовувати функціонал без зайвих ускладнень.

Збільшення задоволення від використання:

Правильний дизайн враховує потреби та очікування користувачів, що призводить до підвищення задоволення від використання ресурсу. Якісний UX дизайн враховує контекст використання та надає користувачеві позитивний досвід.

Зменшення кількості помилок:

Чіткий та добре структурований дизайн допомагає уникнути непорозумінь та помилок користувачів. Логічно розміщені елементи інтерфейсу та зрозумілі підказки зменшують ймовірність неправильного використання функціоналу.

Взаємодія з брендом:

Дизайн інтерфейсу є ключовим елементом бренду, допомагаючи створити консистентність в сприйнятті компанії або проекту. Використання корпоративних кольорів, шрифтів та іконок додає впізнаваності.

Адаптація до різних платформ:

Врахування принципів респонсивного дизайну дозволяє інтерфейсу адаптуватися до різних пристроїв та розмірів екранів, забезпечуючи однаково якісний досвід для користувачів на різних платформах.

Вплив на конверсії:

Правильний дизайн може значно вплинути на конверсії. Візуально привабливий та логічний інтерфейс може покращити конверсійні показники та збільшити взаємодію з ресурсом.

Отже, розробка інтерфейсу є не лише естетичним аспектом, але і стратегічним компонентом успішної веб-розробки, забезпечуючи користувачам приємний, продуктивний та надійний досвід.

Першим екраном нашого додатку буде зустрічати сторінка, з привітанням та можливістю вставити посилання на сайт, для отримання даних для аналізу та рекомендацій.

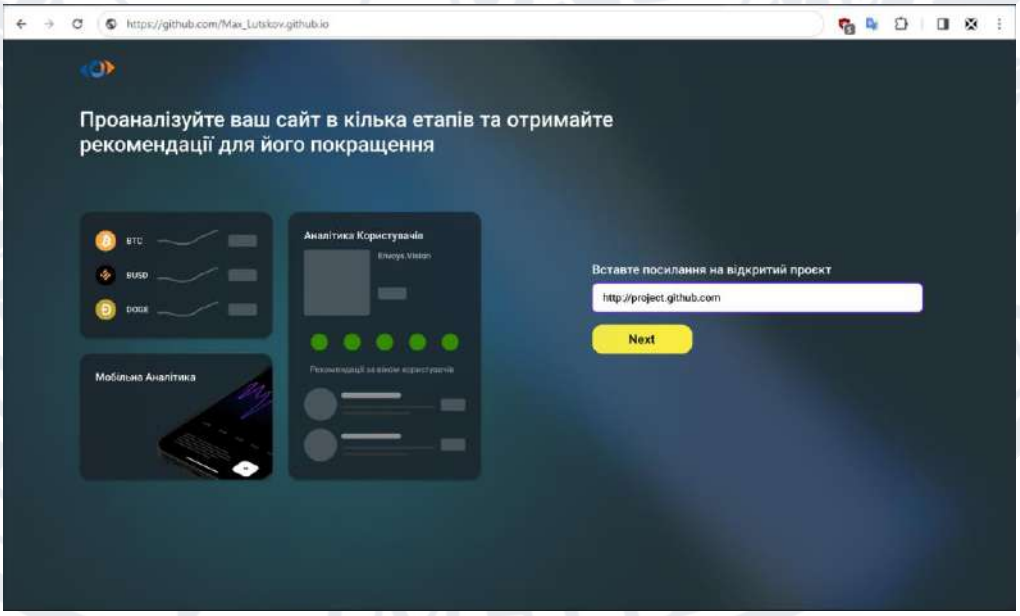


Рисунок 3.1 – Перший екран додатку

Після натискання кнопки «Далі» проводиться не велике завантаження даних, після чого користувач буде перекинутий на сторінку самостійного аналізу.

На поточній сторінці, перед користувачем, буде 4 основні вкладки з даними, які були отримані з його сайту.

На першій вкладці, користувач бачить інформацію, про кількість візитів, кількості переглянутих сторінок, та середній час користувача на його сайті.

Рисунок 3.2 – Статистика сайту

На наступній вкладці, можна проаналізувати, конверсії користувачів, розділення по гендерному признаку та їх вік, це дає змогу підбирати контент під більшу масу користувачів, що в свою чергу буде давати більшу конверсію з вашого сайту.

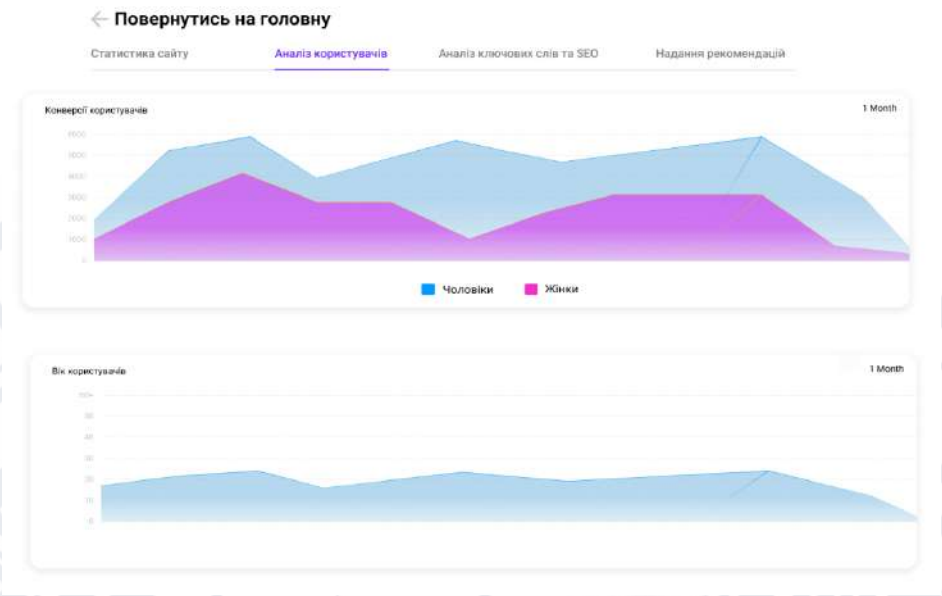


Рисунок 3.3 – Аналіз користувачів

Наступним етапом аналіз ключових слів, завдяки одному з АПІ, він надає рейтинг ключових слів, семантичне ядро, частоту певних слів, саме це дасть змогу покращувати контент, та ключові слова при SEO оптимізації

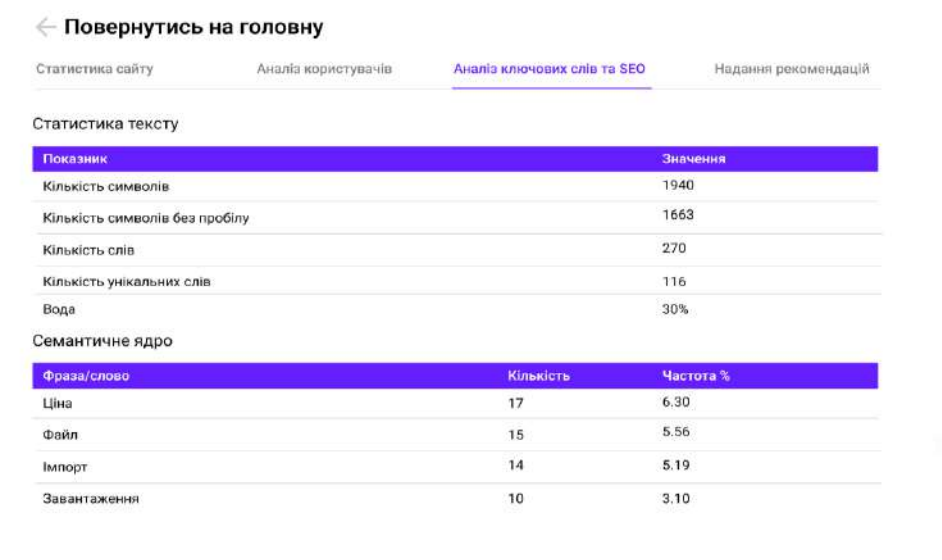


Рисунок 3.4 – Аналіз ключових слів та SEO

Остання вкладка дає користувачу можливість отримати рекомендації від платформи, на базі отриманих даних, які зможуть покращити подальшу роботу

сайту. На ній відображення, поточні проблеми, та прогнозовані покращення, в разі їх вирішення.

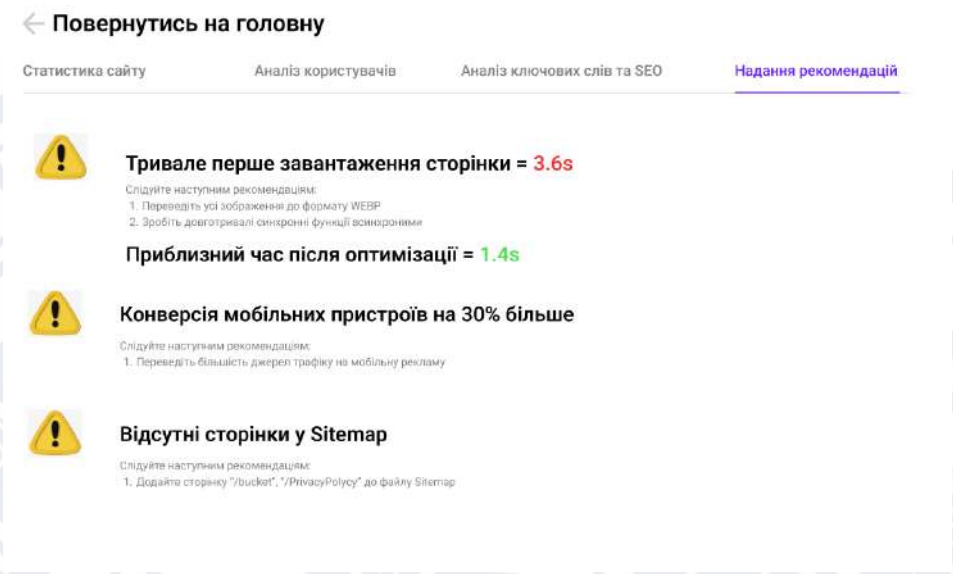


Рисунок 3.5 – Надання рекомендацій

Саме такий правильно побудований інтерфейс є важливою складовою успіху будь-якого веб-додатку. Не лише він робить користування додатком зручним, але також відкриває перед користувачем безліч можливостей для самовдосконалення його веб-сайту. Завдяки цьому інтерфейсу користувач має можливість проводити самостійний аналіз свого сайту, здійснювати контроль над його ключовими параметрами та вчасно реагувати на зміни.

Окрім того, надані додатком рекомендації дозволяють користувачу вдосконалювати свій веб-сайт, враховуючи найновіші тенденції та вимоги галузі. Це створює унікальний шлях для постійного розвитку та оптимізації, забезпечуючи конкурентні переваги в онлайн-середовищі.

Загалом, важливість такого інтерфейсу полягає не лише в полегшенні взаємодії користувача з додатком, але й у створенні умов для постійного росту та вдосконалення веб-простору, що відображається в успішному функціонуванні та конкурентоздатності його веб-сайту.

3.2 Отримання даних для аналізу сайту

Відповідно для аналізу , нам необхідно динамічно отримувати дані з сайту, який ми хочемо проаналізувати, для цього, ми використовуєм наші API , Google Analytics API, SEmRush API та Google insigths. Для початку напишемо запити, які будуть надсилатись до нашого API

```
const apiKey = 'kj345jh8fhg34k76jkhgf35h6f7h356fk7h5g6f73h';
const targetDomain = 'donnu.edu.ua';

const apiUrl = `https://api.semrush.com/?key=${apiKey}&type=domain_organic&export_columns=Ph,Po,Pp,Pd,Nq,Cp,Ur&domain=${targetDomain}`;

fetch(apiUrl)
  .then(response => response.json())
  .then(data => {
    parsedData = JSON.parse(data)
    state.push(parsedData)
    console.log("GETTING Data succsesfull");
  })
  .catch(error => console.error('Помилка запиту:', error));
```

Рисунок 3.6 – Дані SEMRush API

▼ {jsonrpc: "2.0", id: 3,…}
  id: 3
  jsonrpc: "2.0"
▼ result: [{code: "mobile-il", currencies: ["usd", "ils"], isPermitted: true, name: "Mobile Israel",…},…]
  ▼ [0 … 99]
    ▼ 0: {code: "mobile-il", currencies: ["usd", "ils"], isPermitted: true, name: "Mobile Israel",…}
      code: "mobile-il"
      ▶ currencies: ["usd", "ils"]
      isPermitted: true
      name: "Mobile Israel"
      positionsCountOnSERP: 20
      region: "europe"
      searchEngine: "google"
      type: "standard"
    ▶ 1: {code: "af", currencies: ["usd", "afn"], isPermitted: false, name: "Google Afghanistan",…}
    ▶ 2: {code: "ni", currencies: ["usd", "nio"], isPermitted: false, name: "Google Nicaragua",…}
    ▶ 3: {code: "tr", currencies: ["usd", "try"], isPermitted: true, name: "Google Turkey",…}
    ▶ 4: {code: "pk", currencies: ["usd", "pkr"], isPermitted: false, name: "Google Pakistan",…}
    ▶ 5: {code: "mobile-fr", currencies: ["usd", "eur"], isPermitted: true, name: "Mobile France",…}
    ▶ 6: {code: "lu", currencies: ["usd", "eur"], isPermitted: false, name: "Google Luxembourg",…}
    ▶ 7: {code: "ch", currencies: ["usd", "chf"], isPermitted: true, name: "Google Switzerland",…}
    ▶ 8: {code: "mobile-nl", currencies: ["usd", "eur"], isPermitted: true, name: "Mobile Netherland",…}
    ▶ 9: {code: "ua", currencies: ["usd", "uah"], isPermitted: false, name: "Google Ukraine",…}
    ▶ 10: {code: "es", currencies: ["usd", "eur"], isPermitted: true, name: "Google Spain",…}
    ▶ 11: {code: "py", currencies: ["usd", "pyg"], isPermitted: false, name: "Google Paraguay",…}
    ▶ 12: {code: "mobile-au", currencies: ["usd", "aud"], isPermitted: true, name: "Mobile Australia",…}
    ▶ 13: {code: "bo", currencies: ["usd", "bob"], isPermitted: false, name: "Google Bolivia",…}
    ▶ 14: {code: "sn", currencies: ["usd", "xof"], isPermitted: false, name: "Google Senegal",…}
    ▶ 15: {code: "az", currencies: ["usd", "azn"], isPermitted: false, name: "Google Azerbaijan",…}
    ▶ 16: {code: "ht", currencies: ["usd", "htg"], isPermitted: false, name: "Google Haiti",…}
    ▶ 17: {code: "mobile-mx", currencies: ["usd", "mxn"], isPermitted: true, name: "Mobile Mexico",…}
    ▶ 18: {code: "mobile-tr", currencies: ["usd", "try"], isPermitted: true, name: "Mobile Turkey",…}
    ▶ 19: {code: "mx", currencies: ["usd", "mxn"], isPermitted: true, name: "Google Mexico",…}
    ▶ 20: {code: "it", currencies: ["usd", "eur"], isPermitted: true, name: "Google Italy",…}
    ▶ 21: {code: "hr", currencies: ["usd", "hrk"], isPermitted: false, name: "Google Croatia",…}
    ▶ 22: {code: "gr", currencies: ["usd", "eur"], isPermitted: false, name: "Google Greece",…}
    ▶ 23: {code: "na", currencies: ["usd", "nad"], isPermitted: false, name: "Google Namibia",…}
    ▶ 24: {code: "bs", currencies: ["usd", "bsd"], isPermitted: false, name: "Google Bahamas",…}
    ▶ 25: {code: "uk", currencies: ["usd", "gbp"], isPermitted: true, name: "Google UK",…}
    ▶ 26: {code: "mobile-ca", currencies: ["usd", "cad"], isPermitted: true, name: "Mobile Canada",…}
    ▶ 27: {code: "mobile-de", currencies: ["usd", "eur"], isPermitted: true, name: "Mobile Germany",…}
    ▶ 28: {code: "eg", currencies: ["usd", "egp"], isPermitted: false, name: "Google Egypt",…}
    ▶ 29: {code: "mobile-us", currencies: ["usd"], isPermitted: true, name: "Mobile USA",…}
    ▶ 30: {code: "cl", currencies: ["usd", "clp"], isPermitted: false, name: "Google Chile",…}
    ▶ 31: {code: "co", currencies: ["usd", "cop"], isPermitted: false, name: "Google Colombia",…}
    ▶ 32: {code: "mu", currencies: ["usd", "mur"], isPermitted: false, name: "Google Mauritius",…}
    ▶ 33: {code: "tt", currencies: ["usd", "ttd"], isPermitted: false, name: "Google Trinidad and Tobago",…}
    ▶ 34: {code: "am", currencies: ["usd", "amd"], isPermitted: false, name: "Google Armenia",…}
    ▶ 35: {code: "ie", currencies: ["usd", "eur"], isPermitted: true, name: "Google Ireland",…}
    ▶ 36: {code: "mobile-es", currencies: ["usd", "eur"], isPermitted: true, name: "Mobile Spain",…}
    ▶ 37: {code: "bg", currencies: ["usd", "bgn"], isPermitted: false, name: "Google Bulgaria",…}
    ▶ 38: {code: "gh", currencies: ["usd", "ghs"], isPermitted: false, name: "Google Ghana",…}
    ▶ 39: {code: "mz", currencies: ["usd", "mzn"], isPermitted: false, name: "Google Mozambique",…}
    ▶ 40: {code: "pa", currencies: ["usd", "pab"], isPermitted: false, name: "Google Panama",…}
    ▶ 41: {code: "si", currencies: ["usd", "eur"], isPermitted: false, name: "Google Slovenia",…}
    ▶ 42: {code: "fr", currencies: ["usd", "eur"], isPermitted: true, name: "Google France",…}

Рисунок 3.7 – отриманні дані з SEMRush

SEMrush API надає різноманітні дані про органічний та рекламний трафік веб-сайтів. Ось кілька типових параметрів та показників, які можна отримати з використанням SEMrush API:

1. Ph (Keyword): Ключові слова, за якими сайт ранжується в пошукових системах.

2. Po (Position): Позиція сайту в пошукових результатах за конкретним ключовим словом.

3. Pp (Previous Position): Попередня позиція сайту за конкретним ключовим словом.

4. Pd (Position Difference): Різниця між поточною та попередньою позицією.

5. Nq (Volume): Обсяг пошукових запитів для конкретного ключового слова.

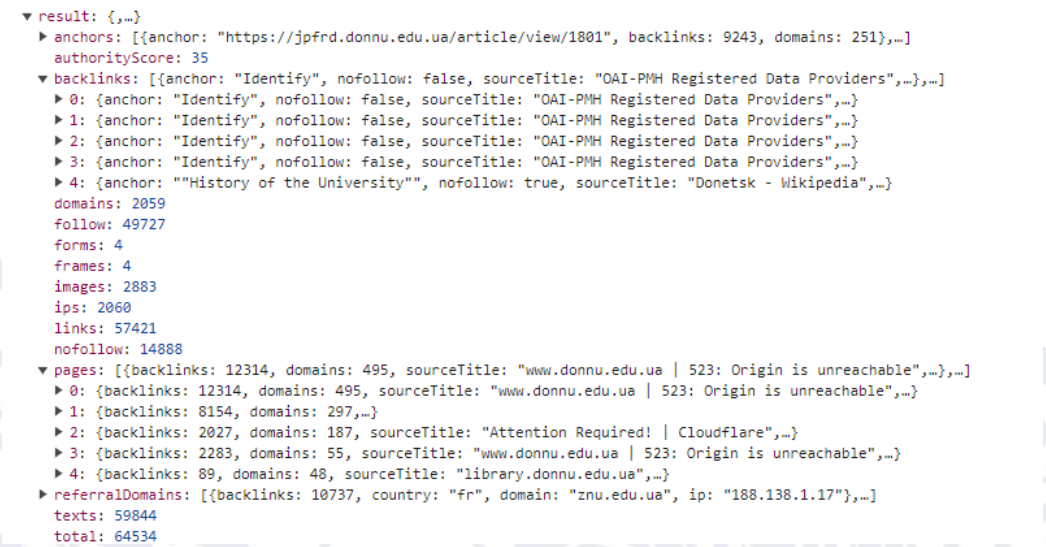
6. Cp (CPC): Вартість за клік на рекламний блок для даного ключового слова.

7. Ur (URL): URL-адреса сторінки, яка ранжується за конкретним ключовим словом.

Ці дані дозволяють вам здійснювати аналіз конкурентоспроможності ваших ключових слів, відстежувати рухи вашого сайту в пошукових системах, а також дізнаватися про рекламні та органічні стратегії конкурентів.

Рисунок 3.8 – Отримання даних з google analytics

Наступні дані – це інформацію з google analytics API



```
▼ result: {,…}
  ▶ anchors: [{anchor: "https://jpfrd.donnu.edu.ua/article/view/1801", backlinks: 9243, domains: 251},…]
    authorityScore: 35
  ▼ backlinks: [{anchor: "Identify", nofollow: false, sourceTitle: "OAI-PMH Registered Data Providers",…},…]
    ▶ 0: {anchor: "Identify", nofollow: false, sourceTitle: "OAI-PMH Registered Data Providers",…}
    ▶ 1: {anchor: "Identify", nofollow: false, sourceTitle: "OAI-PMH Registered Data Providers",…}
    ▶ 2: {anchor: "Identify", nofollow: false, sourceTitle: "OAI-PMH Registered Data Providers",…}
    ▶ 3: {anchor: "Identify", nofollow: false, sourceTitle: "OAI-PMH Registered Data Providers",…}
    ▶ 4: {anchor: ""History of the University"", nofollow: true, sourceTitle: "Donetsk - Wikipedia",…}
    domains: 2059
    follow: 49727
    forms: 4
    frames: 4
    images: 2883
    ips: 2060
    links: 57421
    nofollow: 14888
  ▼ pages: [{backlinks: 12314, domains: 495, sourceTitle: "www.donnu.edu.ua | 523: Origin is unreachable",…},…]
    ▶ 0: {backlinks: 12314, domains: 495, sourceTitle: "www.donnu.edu.ua | 523: Origin is unreachable",…}
    ▶ 1: {backlinks: 8154, domains: 297,…}
    ▶ 2: {backlinks: 2027, domains: 187, sourceTitle: "Attention Required! | Cloudflare",…}
    ▶ 3: {backlinks: 2283, domains: 55, sourceTitle: "www.donnu.edu.ua | 523: Origin is unreachable",…}
    ▶ 4: {backlinks: 89, domains: 48, sourceTitle: "library.donnu.edu.ua",…}
  ▶ referralDomains: [{backlinks: 10737, country: "fr", domain: "znu.edu.ua", ip: "188.138.1.17"},…]
    texts: 59844
    total: 64534
```


Google Analytics – це потужний інструмент веб-аналітики, який забезпечує розширений звітінг та аналіз трафіку на вашому веб-сайті. Ця платформа надає вам детальну інформацію про відвідувачів та їх взаємодію з контентом. Основні категорії даних, які ви можете отримати з Google Analytics, включають:

1. Відвідувачі (Users): Кількість унікальних відвідувачів вашого сайту.

2. Сесії (Sessions): Кількість візитів, здійснених користувачами, включаючи всі їхні взаємодії.

3. Сторінки (Pageviews): Кількість переглядів сторінок на вашому сайті.

4. Середня тривалість сесії (Average Session Duration): Середній час, який відвідувачі проводять на сайті під час одного візиту.

5. Відхід (Bounce Rate): Відсоток візитів, під час яких користувачі залишають сайт після перегляду лише однієї сторінки.

6. Джерела трафіку (Traffic Sources): Інформація про те, звідки приходить трафік на ваш сайт (органічний пошук, прямий трафік, соціальні мережі, реклама тощо).

7. Ключові слова (Keywords): Запити, за якими користувачі знаходять ваш сайт у пошукових системах.

Ці дані дозволяють нам ретельно вивчати поведінку відвідувачів, вдосконалювати контент та стратегії приваблення трафіку для оптимізації веб-сайту користувача.

І третім основним доходом інформації для нас є Google Insigths API

```
const gaApiKey = 'kj345jh8fhg34k76jkhgf35h6f7h356fk7h5g6f73h';
const gaViewId = 'id:383582815';
const psiApiUrl = `https://www.googleapis.com/pagespeedonline/v5/runPagespeed?url=${psiTargetUrl}&key=${psiApiKey}`;

fetch(psiApiUrl)
  .then(response => response.json())
  .then(data => {
    parsedData = JSON.parse(data)
    state.push(parsedData)
    console.log("GETTING Data succsesfull");
  })
  .catch(error => console.error('Помилка запиту до Google PageSpeed Insights:', error));
```

Рисунок 3.9 – отримання даних з google insigts API


```
[["wrb.fr","LsX2he","[[[\"https://www.donnu.edu.ua/uk/\",null,1,1,\"\",null,1,null,
\"gtgyjnseye\",null,\"a3bdcp8ovm\"],[\"https://www.donnu.edu.ua/uk/\",null,2,1,\"\",null,1,
null,\"xct4bq623t\",null,\"a3bdcp8ovm\"],[\"https://www.donnu.edu.ua/uk/\",\"https://www.donnu.
edu.ua\",1,2,null,null,3,[5,\"generic::not_found: record not found for provided key:
\\\"https://www.donnu.edu.ua FF:2\\\"\"],\"efim3bkx4e\",null,\"a3bdcp8ovm\"],[\"https://www.
donnu.edu.ua/uk/\",\"https://www.donnu.edu.ua\",2,2,null,null,3,[5,\"generic::not_found:
record not found for provided key: \\\"https://www.donnu.edu.ua FF:1\\\"\"],\"y9e194z5gu\",
null,\"a3bdcp8ovm\"],[\"https://www.donnu.edu.ua/uk/\",\"https://www.donnu.edu.ua/uk/\",1,3,
null,[[[null,null,2,null,\"https://www.donnu.edu.ua/uk/\"]],[[\"cumulative_layout_shift\",
[null,[[[[[null,null,\"0.00\"],[null,null,\"0.10\"],0.48684491978609673],[[null,null,\"0.10\"],
[null,null,\"0.25\"],0.06951871657754018],[[null,null,\"0.25\"],null,0.4436363636363642]]],
[[null,null,\"0.58\"]]]]],[\"experimental_time_to_first_byte\",[null,[[[[[null,0],[null,800],0.
8586030664395221],[[null,800],[null,1800],0.11158432708688235],[[null,1800],null,0.
029812606473594411]]],[[null,621]]]]],[\"first_contentful_paint\",[null,[[[[[null,0],[null,
1800],0.7474402730375385],[[null,1800],[null,3000],0.1700085324232073],[[null,3000],null,0.
08255119453924932]]],[[null,1840]]]]],[\"first_input_delay\",[null,[[[[[null,0],[null,100],0.
7571333775713337],[[null,100],[null,300],0.19730148197301478],[[null,300],null,0.
04556514045565137]]],[[null,88]]]]],[\"interaction_to_next_paint\",[null,[[[[[null,0],[null,
200],0.559574468085107],[[null,200],[null,500],0.3879432624113479],[[null,500],null,0.
05248226950354614]]],[[null,276]]]]],[\"largest_contentful_paint\",[null,[[[[[null,0],[null,
2500],0.6781052631578993],[[null,2500],[null,4000],0.1414736842105272],[[null,4000],null,0.
18042105263158011]]],[[null,3087]]]]]],null,[[2023,11,19],[2023,12,16]]],2,null,\"abyc2ckiee\",
null,\"a3bdcp8ovm\"],[\"https://www.donnu.edu.ua/uk/\",\"https://www.donnu.edu.ua/uk/\",2,3,
null,[[[null,null,1,null,\"https://www.donnu.edu.ua/uk/\"]],[[\"cumulative_layout_shift\",
[null,[[[[[null,null,\"0.00\"],[null,null,\"0.10\"],0.37859154929577554],[[null,null,\"0.10\"],
[null,null,\"0.25\"],0.03323943661971839],[[null,null,\"0.25\"],null,0.588169014084509]]],
[[null,null,\"1.34\"]]]]],[\"experimental_time_to_first_byte\",[null,[[[[[null,0],[null,800],0.
8426711941143171],[[null,800],[null,1800],0.1269571778909638],[[null,1800],null,0.
03037162799471787 2]]],[[null,649]]]]],[\"first_contentful_paint\",[null,[[[[[null,0],[null,
1800],0.7691583505931109],[[null,1800],[null,3000],0.12483524759932252],[[null,3000],null,0.
10600640180756976]]],[[null,1763]]]]],[\"first_input_delay\",[null,[[[[[null,0],[null,100],0.
9742888402625827],[[null,100],[null,300],0.018052516411378564],[[null,300],null,0.
007658643326039395]]],[[null,5]]]]],[\"interaction_to_next_paint\",[null,[[[[[null,0],[null,
200],0.9476240027748914],[[null,200],[null,500],0.03139091224419022],[[null,500],null,0.
0209850849809227 7]]],[[null,50]]]]],[\"largest_contentful_paint\",[null,[[[[[null,0],[null,
2500],0.5877593755948893],[[null,2500],[null,4000],0.17409099562154703],[[null,4000],null,0.
23814962878355156]]],[[null,3844]]]]]],null,[[2023,11,19],[2023,12,16]]],2,null,\"zktfsmyrsm\",
null,\"a3bdcp8ovm\"]]",null,null,null,"generic"]]
```


Рисунок 3.10 – Отримані дані з google insigts API

Google PageSpeed Insights API надає детальну інформацію про ефективність завантаження веб-сайту. Зокрема, ви можете отримати дані щодо швидкості завантаження на різних пристроях, часу до повної готовності

сторінки, адаптивності для мобільних пристроїв та інших важливих метрик. Це дозволяє вам аналізувати та вдосконалювати продуктивність вашого веб-сайту, сприяючи поліпшенню користувацького досвіду та оптимізації взаємодії з вашим контентом. API допомагає визначити ефективність сайту на основі реальних даних, що включає в себе інформацію про завантаження на різних типах пристроїв, що важливо для забезпечення оптимальної роботи та зручного використання сайту в широкому спектрі умов і пристроїв.

Тепер маючи усі основні данні, ми можемо переходити до їх аналізу, на основі якого будем надавати рекомендації щодо покращення нашого сайту.

3.3 Надання рекомендацій

Для аналізу та надання рекомендацій для веб-сайтів, ми можемо використовувати отримані дані з Google PageSpeed Insights API та інших аналітичних інструментів.

1. Аналіз Даних:

Розглянемо метрики швидкості завантаження, такі як час першого відображення, час до повної готовності та інші.

Визначемо проблемні аспекти, які можуть впливати на продуктивність сайту, такі як великі зображення, неоптимізований код, неефективне використання кешування тощо.

Перший пункт передбачає детальний аналіз метрик швидкості завантаження сторінки, включаючи час першого відображення (First Contentful Paint, FCP), час до повної готовності (Time to Interactive, TTI), та інші. Давайте розглянемо деталізацію цих метрик:

1. First Contentful Paint (FCP):

FCP визначає час, який пройшов від початку навантаження сторінки до моменту, коли користувач бачить перший візуальний елемент. Це важливий показник для оцінки швидкості сприйняття користувачем сторінки.

Формула для розрахунку FCP:

FCP = Час, коли перший візуальний елемент стає видимим

2. Time to Interactive (TTI):

TTI вимірює час, необхідний для того, щоб сторінка стала повністю інтерактивною, тобто користувач може взаємодіяти з усіма елементами сторінки.

Формула для розрахунку TTI:

TTI = Час, коли сторінка стає повністю інтерактивною

3. Визначення Потенційних Проблем:

Визначення потенційних проблем з швидкістю завантаження сторінки за допомогою метрик, таких як First Contentful Paint (FCP) та Time to Interactive (TTI), включає в себе аналіз цих метрик з точки зору користувацького досвіду та оптимізації. Розглянемо приклад:

Допустимо, що аналізуючи дані FCP та TTI для вашого веб-сайту, ви помітили наступні показники:

1. Високий Час FCP:

Замість того, щоб користувачі бачили вміст зараз, перший візуальний елемент відображається через 5 секунд. Це може призвести до враження про те, що сторінка завантажується повільно.

Визначення Проблеми: Високий FCP може вказувати на те, що деякі ресурси, наприклад, великі зображення, завантажуються занадто повільно.

Оптимізація: Зменште розмір зображень, використовуйте формати, такі як WebP, та застосовуйте техніку lazy loading для важких ресурсів.

2. Довгий Час TTI:

Коли сторінка стає інтерактивною, користувачам доводиться чекати більше 8 секунд, перш ніж вони можуть взаємодіяти з елементами.

Визначення Проблеми: Довгий TTI може бути результатом завантаження тяжких скриптів чи блокування основних ресурсів.

Оптимізація: Мінімізуйте та оптимізуйте використання скриптів, розгляньте використання асинхронного завантаження, щоб не блокувати сторінку.

Аналізуючи такі дані та визначаючи проблеми, ви можете приймати конкретні заходи щодо оптимізації веб-сайту, спрямовані на поліпшення швидкості завантаження та забезпечення кращого взаємодій з користувачами.

Ці кроки дозволять нам ідентифікувати та виправити проблеми, пов'язані із часом завантаження вашого веб-сайту, забезпечуючи кращий користувацький досвід та впливаючи на позитивний ранг в пошукових системах.

1. Кешування даних

Використання кешування є важливою стратегією для оптимізації роботи веб-сайту, спрямованою на покращення швидкості завантаження сторінок та зменшення навантаження на сервер. Кешування дозволяє тимчасово зберігати копії ресурсів (зображення, стилі, скрипти тощо) на клієнтських або проміжних серверах, щоб уникнути повторного завантаження їх при кожному новому запиті від користувача. Ось детальніше про використання кешування та його переваги:

1. Типи Кешування:

Кешування на Клієнтському Боці (Client-Side Caching): Клієнт (браузер) зберігає копії ресурсів локально, щоб не завантажувати їх при кожному відвідуванні сторінки. Використовуються заголовки кешування, такі як `Cache-Control` і `Expires`.

Кешування на Серверному Боці (Server-Side Caching): Сервер зберігає копії ресурсів, щоб не генерувати їх заново при кожному запиті. Використовуються проміжні кешувальні сервери, наприклад, Varnish або CDN.

2. Переваги Використання Кешування:

Швидше Завантаження:

Кешування дозволяє швидше відображення сторінок для користувачів, оскільки вони отримують ресурси з локального кешу, замість чекання завантаження з сервера.

Зменшення Навантаження на Сервер:

Серверний кеш дозволяє зменшити обсяг роботи, так як не потрібно генерувати або обробляти ресурси для кожного запиту.

Зменшення Трафіку Мережі:

Клієнтське кешування допомагає зменшити трафік мережі, оскільки ресурси не завантажуються повторно з сервера при кожному переході на нову сторінку.

Підвищення Продуктивності:

Завдяки швидшому завантаженню ресурсів та меншому часу очікування, користувачі отримують покращений досвід використання веб-сайту.

3. Керування Кешуванням:

Оновлення Кешу:

Важливо регулярно оновлювати кеш для відображення актуальної інформації. Це може бути здійснене за допомогою зміни версії файлів або використання заголовків `Cache-Control` для визначення строку дії кешу.

Інвалідація Кешу:

Якщо дані стають застарілими або недійсними, необхідно інвалідувати (очищати) кеш, щоб забезпечити відображення актуальної інформації.

Ми можемо використовувати деякі інструменти та підходи для перевірки наявності та ефективності кешування. Ось деякі з них:

1. HTTP Заголовки:

Заголовки HTTP відповіді сервера можуть містити інформацію про кешування.

Наприклад, заголовок `Cache-Control` може містити директиви, такі як `public`, `private`, `max-age`, які вказують на те, як і де кешувати ресурси.

2. DevTools Браузера:

В інструментах розробника вашого браузера можна перевірити, чи використовується кеш для конкретного ресурсу або сторінки.

Вкладка "Network" дозволяє вам переглядати HTTP-заголовки та перевіряти, чи використовується кеш.

3. Оцінка Часу Завантаження:

Якщо сторінка завантажується швидко, це може бути показником ефективного кешування, оскільки деякі ресурси можуть бути взяті з локального кешу.

4. Логи Сервера:

Аналіз логів сервера може розкрити інформацію про те, які ресурси були видачі сервером, якщо вони були взяті з кешу.

У кожному конкретному випадку важливо вивчати HTTP-заголовки, використовувати інструменти розробника та інші засоби для аналізу та оцінки ефективності кешування на вашому веб-сайті.

Використання кешування допомагає забезпечити швидке та ефективне завантаження сторінок веб-сайту, поліпшуючи загальний користувацький досвід та зменшуючи навантаження на сервери.

Завдяки глибокому аналізу метрик продуктивності веб-сайтів, використанню високоефективних стратегій кешування та впровадженню інших аналітичних інструментів, ми отримуємо цінні інсайти для надання користувачам індивідуальних та стратегічних рекомендацій з покращення їхніх веб-сайтів. Цей комплексний підхід дозволяє нам впливати на різноманітні аспекти веб-досвіду, включаючи швидкість завантаження сторінок, оптимізацію використання ресурсів та підтримку ефективного кешування. Результатом є не лише поліпшений користувацький досвід, а й підвищена конкурентоспроможність веб-сайтів в онлайн-середовищі, що сприяє їхньому успіху та відповідає високим стандартам сучасного Інтернету.

ВИСНОВКИ

В даній науковій роботі було ретельно досліджено та вирішено проблему аналізу та покращення функціональності веб-сайтів через розробку та реалізацію веб-додатку. Головною метою дослідження було створення інструменту, який дозволяє власникам веб-сайтів ефективно збирати дані про роботу своїх проектів та отримувати конкретні рекомендації з метою поліпшення їхньої ефективності.

Для досягнення цієї мети був проведений ретельний аналіз існуючих підходів до аналізу веб-сайтів, а також розглянуті різноманітні інструменти, які можуть бути використані для оптимізації функціоналу веб-ресурсів. Це дозволило визначити ключові аспекти, що впливають на ефективність веб-сайтів та визначити напрямки для подальших досліджень.

Далі в роботі була представлена детальна розробка та реалізація веб-додатку, який охоплює процес збору даних про роботу веб-сайтів, визначення виявлених проблем та систематичне надання конструктивних рекомендацій з їх вирішення. Цей додаток також включає в себе ефективний механізм візуалізації результатів аналізу, що сприяє легшому сприйняттю інформації власниками веб-сайтів.

Отже, наукова робота, зосереджена на створенні інструменту для аналізу та оптимізації веб-сайтів, не тільки вносить вагомий внесок у галузь веб-розробки, але й виходить за межі технічного аспекту, допомагаючи власникам веб-ресурсів ефективно управляти та покращувати свої проекти для досягнення максимального імпакту та задоволення користувачів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. StackOverFlow – найпопулярніший веб-форум [Електронний ресурс] – Режим доступу до ресурсу https://stackoverflow.com/
2. Редактор Sublime Text [Електронний ресурс] – Режим доступу до ресурсу: https://www.sublimetext.com/
3. «Visual Studio Code» - засіб для створення, редагування та зневадження сучасних веб-застосунків і програм для хмарних систем [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Visual_Studio_Code
4. Книга David Herron “Node.js Web Development – 4”
5. React Developing tools [Електронний ресурс] Режим доступу до ресурсу: https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi?hl=ru
6. MVC в Ангуляр [Електронний ресурс] – Режим доступу до ресурсу: https://dev.to/angular/understanding-mvc-services-for-frontend-angular-3e8a
7. Використання роутингу у Vue.js [Електронний ресурс] – Режим доступу до ресурсуhttps://router.vuejs.org/
8. Принцип роботи технології AJAX [Електронний ресурс] – Режим доступу до ресурсу: https://www.hostinger.com.ua/rukovodstva/chto-takoje-ajax/
9. Принцип роботи технології AJAX [Електронний ресурс] – Режим доступу до ресурсу: https://www.hostinger.com.ua/rukovodstva/chto-takoje-ajax/
10. Vuex та використання в розробці [Електронний ресурс] – Режим доступу до ресурсу https://docs.gitlab.com/ee/development/fe_guide/vuex.html
11. Визначення JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/JavaScript
12. Figma — векторний онлайн-сервіс розробки інтерфейсів та прототипування [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Figma

13. Адаптивність та кросбраузерність у веб-розробці [Електронний ресурс] / wikipedia.org – Режим доступу до ресурсу: http://comscienceatschool.blogspot.com/p/22.html

14. API – прикладний програмний інтерфейс [Електронний ресурс] – Режим доступу до ресурсу https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%B8%D0%B9_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B8%D0%B9_%D1%96%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81

15. Server-side rendering [Електронний ресурс] – Режим доступу до ресурсу: https://vuejs.org/guide/scaling-up/ssr.html

16. Методологія БЕМ (Блок Елемент Модифікатор) [Електронний ресурс] – Режим доступу до ресурсу: https://avivi.pro/ua/blog/metodologiya-bem-v-deystvii/

17. Routing Nuxt.js [Електронний ресурс] – Режим доступу до ресурсу: https://nuxtjs.org/docs/get-cypc]

18. HTML – це стандартизована мова розмітки документів для перегляду веб-сторінок у браузері [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/HTML

19. CSS стилі для розмітки. [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/CSS

20. Header та footer як layouts [Електронний ресурс] – Режим доступу до ресурсу: https://nuxtjs.org/docs/directory-structure/layouts/

21. Nuxt.js як основа Server-side rendering [Електронний ресурс] – Режим доступу до ресурсу: https://nuxtjs.org/

22. Для фільтрації використовується інструмент filter [Електронний ресурс] – Режим доступу до ресурсу: https://v2.vuejs.org/v2/guide/filters.html

23. Стів Саудерс "Високопродуктивні веб-сайти: основні знання для фронтенд-інженерів" (2007)

24. Стів Саудерс "Ще швидше веб-сайти: найкращі практики для розробників веб-сайтів" (2009)

25. Ілья Григорік "Високопродуктивне браузерне мережування" (2013)

26. Ніколь Салліван, Ніколас С. Закас "Високопродуктивний CSS: швидше будувати веб-сторінки, виправляючи загальні проблеми CSS" (2011)

27. Гаррі Робертс "Performance First: Посібник для початківців з веб-продуктивності" (2020)

28. Лара Каллендер Хоган "Дизайн для продуктивності: вибір естетики та швидкості" (2014)

29. Стоян Стефанов "Шаблони JavaScript" (2010).