Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему:

«Методи адаптації Web-додатків для використання людьми з обмеженими можливостями»

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконав: студент 2 курсу, групи 1КІ-21м
напряму підготовки (спеціальності)
123 – «Комп'ютерна інженерія»
(шифр і назва напряму підготовки, спеціальності)
Козяр С. О.
(прізвище та ініціали)
Керівник професор кафедри ОТ
к.т.н, Захарченко С. М.
(прізвище та ініціали)
Опонент професор кафедри ЕК
д.т.н., Ліщинська Л. Б.
(прізвище та ініціали)

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.
(прізвище та ініціали)
«9» 12 2022 р

Вінниця ВНТУ 2022

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо-кваліфікаційний рівень: магістр
Спеціальність 123 — «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
обчислювальної техніки
_____ проф., д.т.н. О.Д. Азаров
«15» 09 2022 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

Студенту **Козяру Сергію Олеговичу**

1 Тема роботи. «Методи адаптації web-додатків для використання людьми з обмеженими можливостями», керівник роботи Захарченко Сергій Михайлович, к.т.н., проф. кафедри ОТ затверджено наказом №205-А вищого навчального закладу від 15.09.2022 р.

2 Строк подання студентом роботи 19.12.2022 р.

3 Вихідні дані до роботи —проаналізувати методи і засоби адаптації web-додатків для використання людьми з обмеженими можливостями. Розробити прогрмний продукт на основні проаналізованих методів.

4 Зміст пояснювальної записки: розглянути та проаналізувати методи адаптації web-додатків для використання людьми з обмеженими можливостями; провести порівняння з аналогами; розробка програмного продукту; тестування веб-застосунку та рекомендації користувачу;

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): технічне завдання, домашня сторінка програми. Лістинг основного коду програми.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1— Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1-4	Захарченко Сергій Михайлович	15.09.2022	19.12.2022
5	Небава Микола Іванович	15.09.2022	19.12.2022

7. Дата видачі завдання 15.09.2022 року

8 Календарний план роботи наведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Постановка задач роботи	24.09.2022	вик.
2	Пошук матеріалів про методи покращення доступності веб-застосунків	30.09.2022	вик
3	Аналіз методів покращення доступності	5.10.2022	вик
4	Аналіз сучасних інструментів для створення веб-застосунків	20.10.2022	вик
5	Підготовка матеріалів та опис веб-застосунку	27.10.2022	вик
6	Підготовка економічної частини	10.11.2022	вик
7	Оформлення пояснювальної записки	20.11.2022	вик
8	Перевірка якості виконання магістерської кваліфікаційної роботи та усунення недоліків	19.12.2022	вик

Студент _____ Козяр Сергій Олегович

(підпис)

Керівник роботи _____ к.т.н., проф. каф ОТ Захарченко Сергій Михайлович

(підпис)

АНОТАЦІЯ

УДК 004.51

Козяр С.О. Методи адаптації web-додатків для використання людьми з обмеженими можливостями. Магістерська кваліфікаційна робота зі спеціальності 123 — Комп'ютерна Інженерія, Вінниця: ВНТУ, 2022, 132 с.

На укр.мові. Бібліогр.: 34 назв; рис.: 33 ; табл. 12

Дана магістерська робота методам адаптації web-додатків для використання людьми з обмеженими можливостями. Поняття web-достуності набирає популярність через зростання кількості різноманітних Інтернет-сервісів, які все більше є нівд'ємною частиною життя людей.

У роботі проведено аналіз методів покращення доступності web-додатків для можливості освоєння web-ресурсів людьми з обмеженими можливостями. В роботі проведений аналіз сучасних підходів до вирішення задач. Розроблено web-додаток з дотриманням принципів покращення достуності сайтів.

В даній роботі були виконані економічні розрахунки, які дозволяють визначити доцільность розробки нового програмного продукту.

Ключові слова: web-додаток, доступність, сайт.

ANNOTATION

Kozyar S.O. Methods of adapting web applications for use by people with disabilities. Master's thesis on specialty 123 — Computer Engineering, Vinnytsia: VNTU, 2022, 132 p.

In the Ukrainian language. Bibliography: 34 titles; Fig.: 33; table 12

This master's thesis deals with methods of adapting web applications for use by people with disabilities. The concept of web-friendliness is gaining popularity due to the growing number of various Internet services, which are increasingly an integral part of people's lives.

The paper analyzes the methods of improving the accessibility of web applications for the possibility of mastering web resources by people with disabilities. The paper analyzes modern approaches to problem solving. The web application was developed in compliance with the principles of improving the accessibility of sites.

In this paper, economic calculations were performed that allow us to determine the feasibility of developing a new software product.

Keywords: web application, accessibility, site.

ЗМІСТ

ВСТУП

Інтернет загалом стає дедалі найважливішим ресурсом в багатьох аспектах нашого життя, включаючи: освіту, роботу, уряд, торгівлю, охорону здоров'я, відпочинок тощо. Важливо, щоб Інтернет був доступним для всіх, щоб забезпечити рівний доступ і рівні можливості для людей з обмеженими можливостями. Доступний Інтернет може допомогти людям з обмеженими можливостями брати більш активну участь у житті суспільства.

Що таке веб-доступність? Веб-доступність або цифрова доступність – це здатність програми, веб-сайту чи програмного забезпечення бути доступними для людей з обмеженими можливостями, які впливають на їхній зір, слух, когнітивні та/або рухові функції [1].

Мережа в основному розроблена для роботи для всіх людей, незалежно від їх апаратного забезпечення, програмного забезпечення, мови, місця розташування чи здібностей. Коли Інтернет досягає цієї мети, він стає доступним для людей із різноманітним діапазоном слуху, рухів, зору та когнітивних здібностей.

Таким чином вплив інвалідності радикально змінюється в Інтернеті, оскільки Інтернет усуває перешкоди для спілкування та взаємодії, з якими стикаються багато людей у фізичному світі. Однак, коли веб-сайти, програми, технології чи інструменти розроблені погано, вони можуть створювати перешкоди, через які люди не зможуть користуватися Інтернетом.

Виходячи із розглянутого, завдання подальшого вдосконалення доступності сайтів є **актуальною задачею**.

Метою дослідження магістерської роботи є методи покращення доступності сайтів.

Задачі дослідження магістерської роботи:

— здійснити огляд стану роботи над доступністю сайтів у світі

— розглянути та проаналізувати стандарти та методи покращення доступності сайтів;

— створити web-додаток і досягти високого рівня доступності шляхом оглянутих методівпокращення доступності.

Об’єкт дослідження магістерської роботи — процес покращення доступності сайтів.

Предмет дослідження магістерської роботи — методи покращення доступності сайтів для людей з обмеженими можливостями.

Методи дослідження магістерської роботи: використовувались методи статистичного аналізу для визначення потужності вибірки потенційних споживачів, методи критеріального аналізу для формування узаагальненого критерія доступності сайту.

Наукова новизна отриманих результатів магістерської роботи полягає у тому, що:

— удосконалено методи покращення доступності сайтів для людей з обмеженими можливостями.

Практичне значення одержаних результатів магістерської роботи:

— розроблено додаток із функціоналом інтернет магазину, який є адаптований під потреби людей із обмеженими можлвостями.

Апробація Результати роботи опубліковані у матеріалах LI регіональної науково-технічної конференції ВНТУ.

1 ОГЛЯД СТАНУ ДОСТУПНОСТІ У МЕРЕЖІ ІНТЕРНЕТ

1.1 Важливість покращення доступності web-ресурсів

Функція веб-доступності забезпечує повний доступ до веб-сайту для всіх користувачів, незалежно від рівня здібностей, стану чи обставин користувача. Веб-доступність зосереджена на усуненні бар'єрів, які можуть заблокувати доступ користувача до веб-сайту, і застосуванні активного підходу до створення повністю інклюзивного веб-сайту для всіх, включаючи людей із вадами зору, когнітивними, фізичними та слуховими вадами [2].

Веб-доступність — це не лише етика, але й розумна бізнес-практика. З економічного погляду ігнорування спільноти інвалідів означає ігнорування мільйонів потенційних клієнтів із мільярдною купівельною спроможністю. Тільки в Сполучених Штатах купівельна спроможність користувачів Інтернету з допоміжними технологіями становить понад 350 мільярдів доларів [2].

Розглянувши розмір цієї спільноти людей з обмеженими можливостями в решті Америки, Європи та інших регіонів, стає зрозуміло, що ігнорувати цю демографію було б помилкою. У багатьох місцях веб-доступність захищена різними законами. [1]

Це поняття, яке дозволяє забезпечувати рівний доступ кожному, особливо людям з інвалідністю. Люди з інвалідністю мають нижчий рівень зайнятості ніж рівень осіб без інвалідності, згідно зі звітами створеного бюро статистики праці (BLS). Для всіх вікових груп BLS виявив, що у 2015 році коефіцієнт зайнятості населення був нижчим для осіб з інвалідністю ніж для тих, хто не має інвалідності.

Для цілей цього аналізу особи класифікуються як зайняті безробітні (непрацюючі особи, які активно шукають роботу), або не є робочою силою (осіб без роботи, які не шукають роботу активно). Цей показник показує, що в середньому рівень інвалідності є вище серед осіб з нижчим рівнем освіти і що люди з обмеженими можливостями мають нижчий рівень працевлаштування, ніж особи, які не мають інвалідності [2]. Порівняно нижчий рівень зайнятості

осіб з обмеженими можливостями в цілому відображає, що рівень зайнятості загалом нижчий для осіб з меншою освітою та нижчий рівень зайнятості людей з інвалідністю в межах кожного рівня освітнього рівня. За цим показником особи класифікувалися як такі, що мають одну або більше видів інвалідності, якщо вони повідомили про будь-яку з цих характеристик:

— глухота або серйозні труднощі слухання;

— сліпота або серйозні проблеми із зором при носінні окулярів;

— серйозні труднощі з концентрацією, запам'ятовування або прийняття рішень через фізичні, психічний або емоційний стан;

— серйозні труднощі при ходьбі або підйом по сходах; труднощі з одяганням або купанням;

— труднощі з виконанням доручень поодинці, наприклад відвідування лікаря.

Кількість людей віком від 25 до 64 з інвалідністю була більшою у 2015 році, ніж у 2010. Певною мірою ця зміна відображає зростання населення між 2010 і 2015 роками. Вищий відсоток людей похилого віку мали інвалідність порівняно з молодшими людьми у 2015 році. Наприклад, рівень інвалідності становив 15 % для людей віком від 55 до 64 років, у порівнянні з 10 відсотками для людей віком від 45 до 54 років, 6 %для людей віком від 35 до 44 років і 4 відсотки для людей віком від 25 до 34 років. Рівень інвалідності серед людей віком від 25 до 34 років був вищим у 2015 році (4,2 відсотка), ніж у 2010 році (3,7 відсотка) [3]. Зібрані данні у повному обсязі зображені на рисунку 1.1.

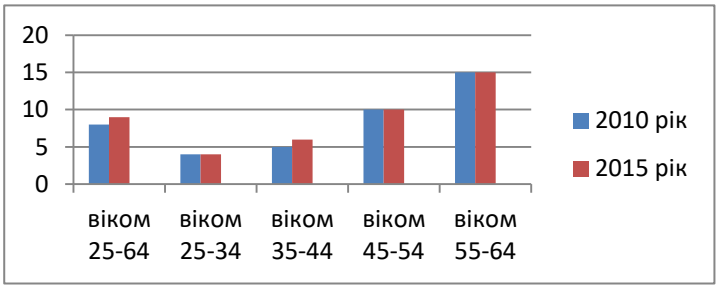

Рисунок 1.1 — Відсоткова статистика про кількість людей із вадами

У 2015 році рівень інвалідності був вищим для осіб з нижчою освітою, ніж у тих, хто має вищу освіту, так і в кожній віковій групі. Рівень інвалідності становив 16% для людей віком від 25 до 64 років, що не закінчили середню школу, порівняно з 11% для тих, хто закінчив середню школу, 10% для тих, хто закінчив якийсь коледж, 8 % для тих хто має ступінь молодшого спеціаліста, 4 % для тих, хто має а ступінь бакалавра та 3% для тих, хто має ступінь магістра або вищий ступінь. Опираючись на ці дані можна сказати те, що людям з обмеженими можливостями і з зростанням складності навчання, важко продовжувати навчання і отримувами освіту вищих рівнів. Ці моделі загалом спостерігалися у кожній віковій групі, за деякими винятками. Групи найвищих освітніх досягнень більші для найстаршої групи (від 55 до 64 років), ніж у наймолодшої групи (від 25 до 34 років). Зокрема, серед людей віком від 55 до 64 років, рівень інвалідності становив 23% вище для осіб, які не закінчили середню освіту 29%, ніж для тих, хто має ступінь магістра або вище ступеня 6%. Навпаки, серед віком від 25 до 34 років рівень інвалідності був на 6% вищим для тих, хто не закінчив середню школу 7% ніж для тих, хто закінчив магістратуру або вище ступінь 1%. У той час як рівень інвалідності в цілому вище для літніх людей, ніж для молоді, проблеми за освітнім рівнем у кожній віковій групі однаково великі [3]. Оглянути повну статистику можна на рисунку 1.2.

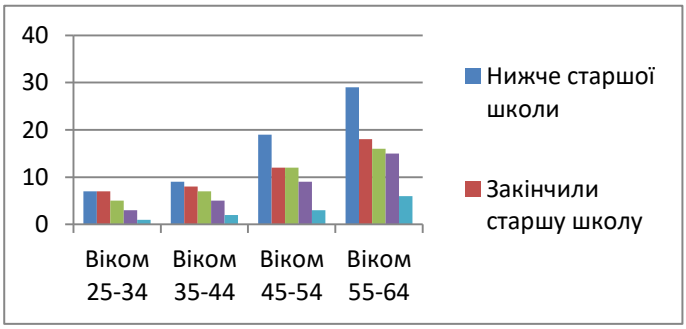

Рисунок 1.2 — Статистика із рівнями освіти людей з обмеженими можливостями

У 2011 році Національний центр статистики освіти виявив, що 11% студентів бакалаврату повідомили про наявність інвалідності, а останні дані показали, що майже 13% населення США мають інвалідність. Потенційне виключення такого значного відсотка населення суперечило б місії UNCG бути «інклюзивною, співпрацюючою та чуйною установою, яка змінює життя студентів і громад, які вона обслуговує».

Дев'ятнадцять відсотків студентів від 2015 до 2016 років повідомили про наявність інвалідності. У роках від 2015 до 2016 відсоток студентів, які повідомили про інвалідність, становив 19% для студентів і 20% для студенток. Були певні відмінності у відсотках студентів з обмеженими можливостями за такими характеристиками, як статус ветерана, вік, статус утриманця. Наприклад, 26% студентів, які були ветеранами, повідомили про наявність інвалідності, порівняно з 19 відсотками студентів, які не були ветеранами. Відсоток студентів з обмеженими можливостями був вищим серед осіб віком 30 років і старше (23 відсотки), ніж серед людей віком від 15 до 23 років (18 відсотків) [4]. Оглянути цю статистику можна на таблиці 1.1.

Таблиця 1.1 — Статистика національного центру статистики освіти

	Бакалаври		Магістри	
	Студенти із вадами	Студенти без вад	Студенти із вадами	Студенти без вад
Всього	19.4	80.6	11.9	88.1
Чоловіки	19.2	80.8	9.9	90.1
Жінки	19.6	80.4	13.3	86.7
15 to 23	17.6	82.4	8.1	91.9
24 to 29	21.6	78.4	11.3	88.7
30 або старші	22.6	77.4	13.5	86.5
Ветеран	25.8	74.2	17.1	82.9
Не ветеран	19.1	80.9	11.6	88.4

Оглянувши цю статистику, можна зробити висновок, що кожеш п’ятий студент в середньому має, певні проблеми із здоров’ям. Ці проблеми в свою чергу можуть вплинути на його навчально-робочий процес. У наш період комп’ютеризації всього, робота над доступністю у навчальних Інтернет джерелах, сайтіах із методичками, особистих web-кабінетах студентів та навіть у сайті самого вищого учбового закладу, може полегшити роботу студентів та зробити її більш комфортною, що в свою чергу вплине на їх успішність та продуктивність. А найголовніше ці студенти зможуть відчути максимальну допомогу і турботу від своїх учбових закладів, що також часто може вплинути на самопочуття та продуктивність студента. Після прикладу із освітою, слід глянути на статистику по працевлаштуванню людей із обмеженими можливостями.

Дослідження BLS показали, що люди з обмеженими можливостями, мають набагато менший відсоток працевлаштованості, ніж особи без інвалідності. У 2015 році серед кожної обстеженої вікової групи по цьому показнику відсоток зайнятості був вищим для осіб без інвалідності, ніж для людей з обмеженими можливостями. Цей розрив коливався від 43 % для людей віком від 25 до 34 років до 53 % для людей віком від 45 до 54 років. Серед осіб з інвалідністю вищий відсоток людей віком від 25 до 34 років були працевлаштовані (35 відсотків), ніж людей віком від 35 до 44 років (29 відсотків), людей віком від 45 до 54 років (28 відсотків) та від 55 до 64 років (24 відсотки). Структура зайнятості за віковими групами відрізнялась для людей без інвалідності. Хоча відсоток людей віком від 25 до 34 років, які були зайняті (78 відсотків) був вищим, ніж відсоток для у віці від 55 до 64 років (69 відсотків) він був нижчим, ніж для людей віком від 35 до 44 років і від 45 до 54 років (обидва 81 відсоток) [4]. Статистика із рівнем працевлаштованості людей забражена на рисунку 1.4.

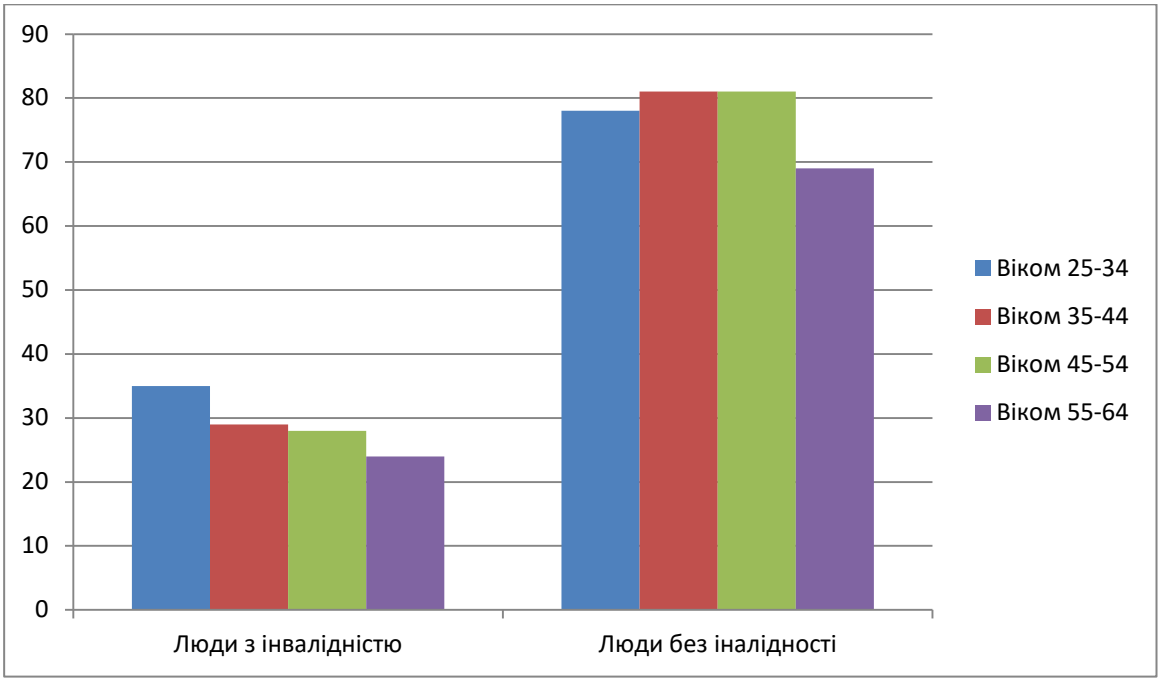

Рисунок 1.3 — Статистика працевлаштованості людей у 2015 році

У 2015 році низький рівень освіти був пов'язаний з нижчим відсотком зайнятості як для осіб з інвалідністю так і без. Серед інвалідів віком від 25 до 64 років, відсотки працевлаштування тих, хто не закінчив середню школу (15 відсотків) або закінчили лише середню школу (22 відсотки). нижче, ніж для тих, хто закінчив коледж (31 відсоток), вчений ступінь (35 відсотків) або ступінь бакалавра або вище (45 відсотків). Так само серед осіб без інвалідності, відсоток зайнятості для тих, хто не закінчив середню школу (62 відсотки) або закінчили лише середню школу (73 відсотки). нижче, ніж для тих, хто закінчив якийсь коледж (76 відсотків), ступінь молодшого спеціаліста (82 відсотки) або ступінь бакалавра і вище (84 відсотки). Розрив в відсотках зайнятості між тими, хто має і не має інвалідність була меншою для тих, хто отримав ступінь бакалавра або вищий ступінь (39 відсоткових пунктів), ніж для осіб з науковим ступінь (47 відсотків), з свідоцтвом про середню освіту цей рівень становить (51 відсотків) і для тих, хто не закінчив середню школу (47 відсотків) [4]. Статистика про рівень освіти людей показана на рисунку 1.4.

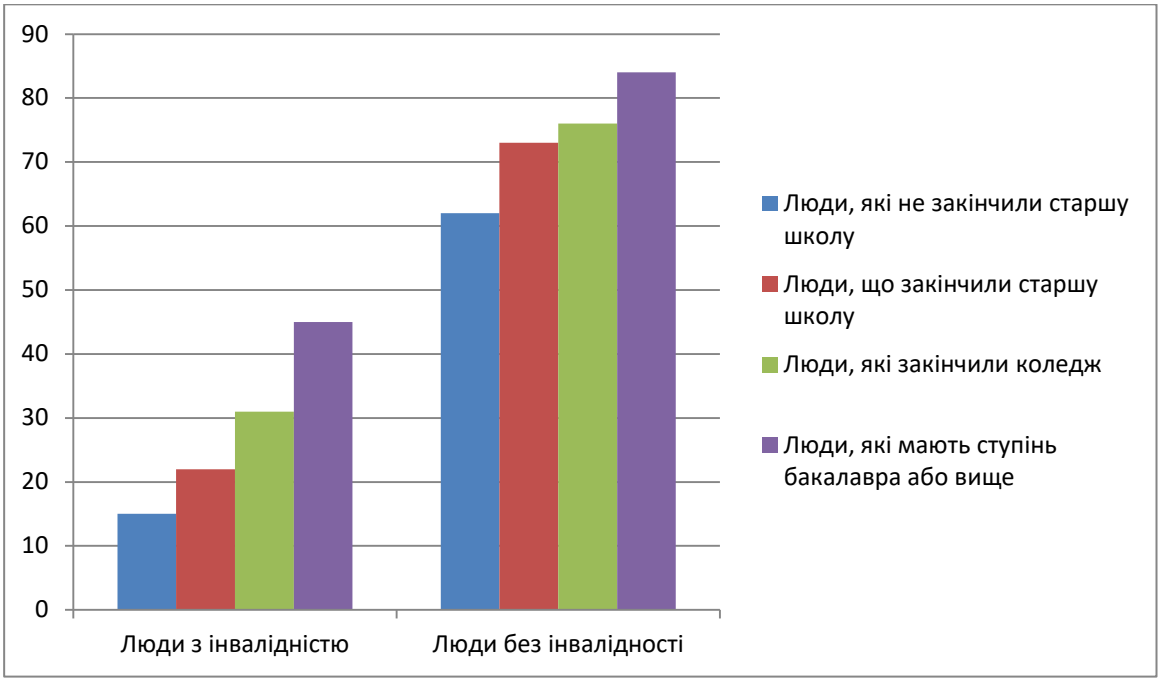

Рисунок 1.4 — Статистика про рівень освіти людей

У 2015 році відсоток людей з обмеженими можливостями віком 25-64 років, які були безробітними 3,4% відрізняється від відсотка безробіття тих, хто не має інвалідності 3,6% відсотка; проте там були відмінності за освітнім рівнем. Важливо мати на увазі, відсоток зайнятості людей віком від 25 до 64 років з інвалідністю нижчий , ніж для тих, хто не має обмежених можливостей. Отже, чисельність безробітних осіб відносно зайнятих осіб (тобто безробіття за визначенням BLS) вищий для людей віком від 25 до 64 років з обмеженими можливостями 11%, ніж для осіб без інвалідністі 4,5%. Для осіб без інвалідності, вища освіта досягнення часто асоціювалося з нижчим рівнем безробіття відсотки. Наприклад, тих, хто закінчив ступінь доцента та ті, хто закінчив а бакалавра або вищого ступеня мали нижчий рівень безробіття відсотків, ніж у тих, хто не завершив високий рівень школа. Серед тих, хто не закінчив високі школи, відсоток безробіття для осіб с інвалідності (2,4 відсотка) була нижчою, ніж у осіб без інвалідності (6,1 відс.). Навпаки, серед тих які отримали ступінь бакалавра або вищу ступінь, відсоток безробіття був вищим для осіб с інвалідів (3,5 відсотка), ніж для тих, хто не має інвалідності (2,0 відсотка). Повна статисктика зображена на риснку 1.5.

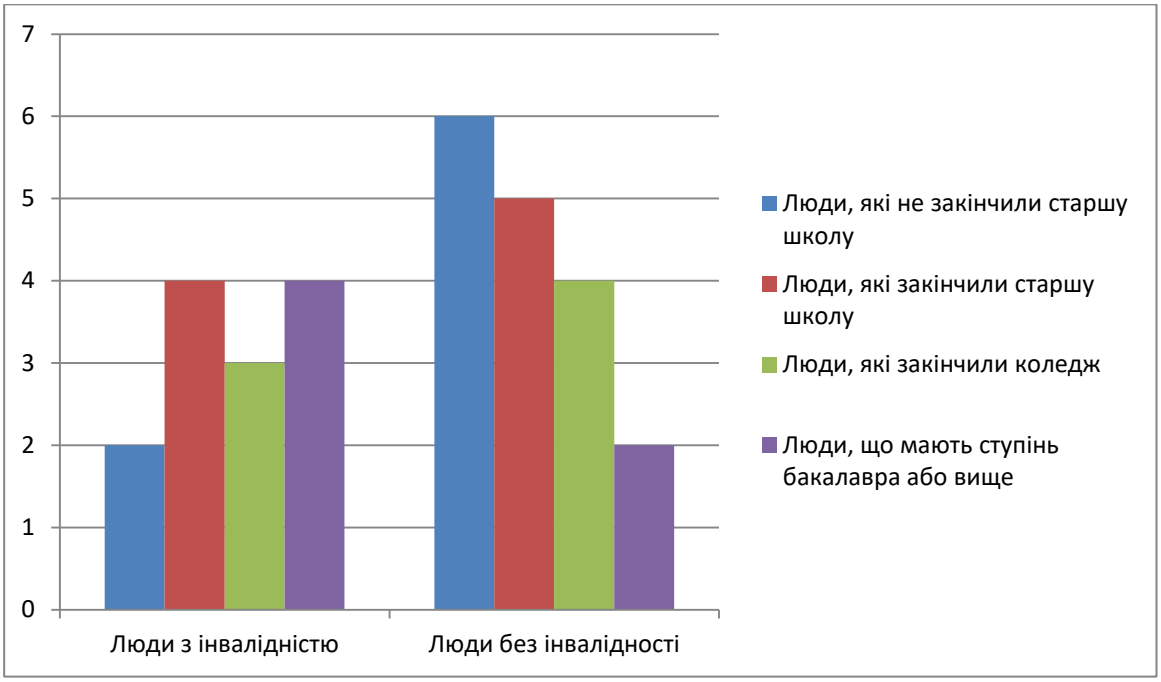

Рисунок 1.5 — Статистика безробітніх людей відносно їх рівня освіти

З цього можна зробити висновки, що люди з обмеженими можливостями у більшості прагнуть влаштуватися на роботу, де потрібен високий рівень освіти та кваліфікації, тому що роботи для людей без освіти в більшості пов'язані із фізичною працею, якою люди з інвалідністю займатися не можуть. Робота над доступністю в Інтернеті позитивно вплине на цю статистику, через спрощення та покращення умов у багатьох аспектах людям буде легше влаштуватись на роботу і в свою чергу це зробить робочий процес набагато комфортнішим для них [4].

1.2 Переваги роботи над доступністю

Якщо говорити про переваги роботи над доступністю для бізнесу, то їх також чимало:

Покращення зручності використання продукту. Доступність впливає на зручність використання. Коли при розробці враховується користування продуктом для людей із крайнім ступенем інвалідності, ми зазвичай приносимо користь багатьом іншим користувачам із менш важкими недугами, тимчасовими обмеженнями, хворобами, віком чи соціальними факторами.

Наприклад, вищий колірний контраст може допомогти мобільним користувачам, незалежно від того, перебувають вони в яскравому Лас-Вегасі чи вночі в пустелі Невада. Чистий простий дизайн може принести користь усім користувачам, а не лише деяким із когнітивними вадами. Siri від Apple спочатку була створена для людей із втратою зору, але тепер широко використовується й зрячими людьми. Статистика: згідно з дослідженням ек ономічного впливу Forrester Research, проведеним на замовлення Microsoft, із 319 компаній 252 погодилися, що інклюзивне програмне забезпечення покращує зручність використання та покращує взаємодію з клієнтами [5];

Підвищення рентабельності інвестицій. Фінансово доцільно охопити ще від 15 до 20% людей. Звичайно, доступність ваших веб-сайтів або програм пов'язана з витратами; наприклад, проведення аудитів, навчання або додатковий маркетинг. Однак переваги зазвичай переважують ці витрати. Статистика. Модель для розрахунку ROI можна знайти в книзі Cost-Justifying Usability. Приклад розрахунку призводить до зростання ринку на 8% і рентабельності інвестицій 2,4 до 1. Цікаво, що британський супермаркет Tesco виявив, що доступність їхнього онлайн-сайту та інших цифрових продуктів призвела до річного збільшення продажів на 13 мільйонів фунтів стерлінгів. Крім того, згідно з опитуванням Click-Away Pound;

Підвищення рейтингу пошукової системи. Індексний бот Google — це схожий на користувача з вадами зору та мільярдом друзів. Він покладається на текст для розуміння зображень і медіаконтенту. Коли веб-сайти надають корисний альтернативний текст зображенням, аудіо та відео, це полегшує пошук сайту, підвищує його рейтинг і отримує рекомендації. Статистика: у щотижневому мовленні NPR, яке щотижня відвідує близько 2,1 мільйона слухачів, надання стенограм збільшило пошуковий трафік на 6,86% [5];

Репутація. Окрім юридичних наслідків, закони у багатьох країнах щодо доступності також можуть значно вплинути на репутацію бізнесу. Тільки подумайте про це: в очах багатьох людей Domino's, ймовірно, назавжди запам'ятається як бренд, який виступав проти основних прав сліпої людини.

На щастя, подібно до того, як погані методи доступності шкодять вашій репутації, хороші можуть справді допомогти. Якщо ви докладаєте зусиль, щоб справді обслуговувати всіх своїх клієнтів і користувачів, це зробить ваш бренд помітним не лише в очах користувачів, які безпосередньо виграють від доступності, але й тих, хто найбільше цінує соціальний вплив. Як зручно, дві аудиторії, які найбільше цінують прагнення бренду робити речі правильно, саме ті, які складають найбільшу частку глобальної споживчої бази: міленіали. Оскільки ці клієнти є тими, хто, най імовірніше, відмовляться від бренду через виявлену погану практику у цьому аспекті, надання нерівного доступу потенційно означає втрату значного відсотка клієнтів.

1.3 Типи людських вад та проблеми з якими стикаються люди в інтернеті

У світі надзичайно багато людей з обмеженими можливостями і людські вади бувають різні. Тому варто розглянути усі можливі види вад, які заважають людям користуватись Інтернетом та розглянути з якими саме проблемами стикаються люди.

Сліпота — це стан важкого порушення зору, який обмежує здатність людини бачити предмети в різних умовах і факторах зовнішнього середовища. Повністю сліпа людина страждає від повної втрати зору і може не бачити навіть при світлі [5]. Частково сліпі люди мають слабкий або обмежений зір, який неможливо виправити за допомогою окулярів або лінз. Визначення сліпоти включає широкий спектр порушень зору, таких як:

— дальтонізм;
— низька або знижена контрастна чутливість;
— слабкий зір (неможливо розрізнити рухомі фігури);
— повна сліпота (неможливість відрізнити світло від темряви).

Критерії юридичної сліпоти відрізняються в різних країнах. У США людина вважається сліпою, якщо її гострота зору становить 20/200 або менше,

з коригувальними лінзами або без них. Людина із зором 20/200, стоячи на відстані 20 футів від очної карти, може бачити те, що людина з нормальним зором може бачити на відстані 200 футів. Особи з тунельним зором, чиє поле зору становить 20 градусів у діаметрі або менше, також вважаються юридично сліпими.

Центр контролю та профілактики захворювань (CDC), прогнозує, що до 2050 року число людей із проблемами зору збільшиться більш ніж удвічі через збільшення вікових проблем із очима та інших хронічних захворювань, таких як діабет.

Поява смартфонів і планшетів із доступними додатками дозволила більшості незрячих людей, які переглядають Інтернет, легше отримати доступ до онлайн-сервісів. Точної статистики щодо використання Інтернету незрячими людьми немає. WebAIM провів опитування серед людей з вадами зору, яке показало, що 70% з них мають високий рівень володіння Інтернетом. Дивно, але ті, хто мав серйозніші проблеми із зором, повідомили про вищий рівень навичок.

Інтернет — це платформа, яка дозволяє брати участь у громадських заходах, використовувати бізнес-можливості, ділитися знаннями та багато іншого. Люди з вадами зору мають однакові потреби в доступі до цієї інформації. У деяких випадках сліпі люди більше покладаються на онлайн-сервіси, ніж на фізичне пристосування.

У таких ситуаціях, як пандемія COVID-19, коли діють правила соціального дистанціювання, можливість для незрячих людей користуватися онлайн-сервісами стає вирішальною. Зрештою, після того, як пандемія мине, Інтернет усе ще залишатиметься основним способом спілкування, що дозволить працювати вдома, робити покупки в Інтернеті, банківські послуги тощо. Це робить сильний акцент на доступності Інтернету та підтримці допоміжних технологій. Нещодавно WebAIM провів автоматичне сканування одного мільйона найпопулярніших сторінок, щоб перевірити функції доступності. Результати показали, що лише частина веб-сайтів відповідала

критеріям доступності. Одним із головних висновків було те, що більшість веб-сайтів надають незрозумілі мітки для елементів сторінки. Замість того, щоб отримати значущу семантичну мітку, багато зображень, кнопок, пунктів меню позначаються як «image1», «button1» тощо. Коли програми зчитування з екрана читають такі загадкові описи незрячим користувачам, це створює лише плутанину та проблеми під час навігації веб-сайтом .

Інші цифрові бар'єри включають несумісність із програмами зчитування з екрана, складні макети, використання зображень і графіки замість тексту, сторінки, які стають недоступними через оновлення програмного забезпечення чи додавання нового вмісту тощо. Здебільшого онлайн-контент зосереджений на візуальній презентації, оскільки дизайнери та розробники надають перевагу створенню привабливих інтерфейсів, які привертають увагу користувача. Однак у більшості випадків це робить Інтернет для сліпих і людей із вадами зору недоступним і недоступним.

Іншою важливою проблемою є вартість допоміжних технологій, які багато людей із вадами зору просто не можуть собі дозволити. Візьмемо, наприклад, брайлівську клавіатуру, вартість якої коливається від 3500 до 15 000 доларів. Подібним чином комерційні пакети з інтегрованим програмним забезпеченням для зчитування екрана та диктування також є досить дорогими. Багато розробників, які не розуміють і не знають, як сліпі люди переглядають Інтернет, додають на веб-сайт функції, які є візуально привабливими, але не мають жодної цінності для людей із вадами зору. Наступні аспекти спричиняють труднощі для людей із вадами зору в доступі до онлайн-контенту:

— зображення без позначок, якщо посилання або кнопка представлені лише зображеннями без тексту, програма зчитування з екрана не зможе розшифрувати з них жодну значиму інформацію, і користувач не матиме уявлення про їхнє призначення;

— недоступна CAPTCHA хоча це важливий інструмент для захисту від спаму та непотрібного трафіку, він не підходить для людей із вадами зору;

— деякі веб-сайти надають звукові описи для зображень CAPTCHA, але якість звуку зазвичай не дуже добра;

— складні макети сторінок, тобто занадто захаращені або складні макети сторінок можуть заплутати під час навігації за допомогою програми зчитування з екрана;

— рухомий текст/зображення, тобто текст, поданий за допомогою зображень або рухомих слайд-шоу, також створює проблеми, якщо він не супроводжується альтернативним текстом;

— сповіщення про файли cookie, а саме банери з файлами cookie безпосередньо на початку веб-сайту можуть створювати перешкоди, якщо вони не мають відповідних міток і елементів керування для незрячих користувачів.

За даними WHO, у світі налічується 285 мільйонів людей, які через певну інвалідність (тобто страждають слабким зором) не можуть читати весь вміст веб-сайту. 39 мільйонів із них сліпі й не можуть отримати доступ до вмісту через зір. Точнішу статистику кількості людей із проблемами зору у певних світових регіонах зображена на рисунку 1.6.

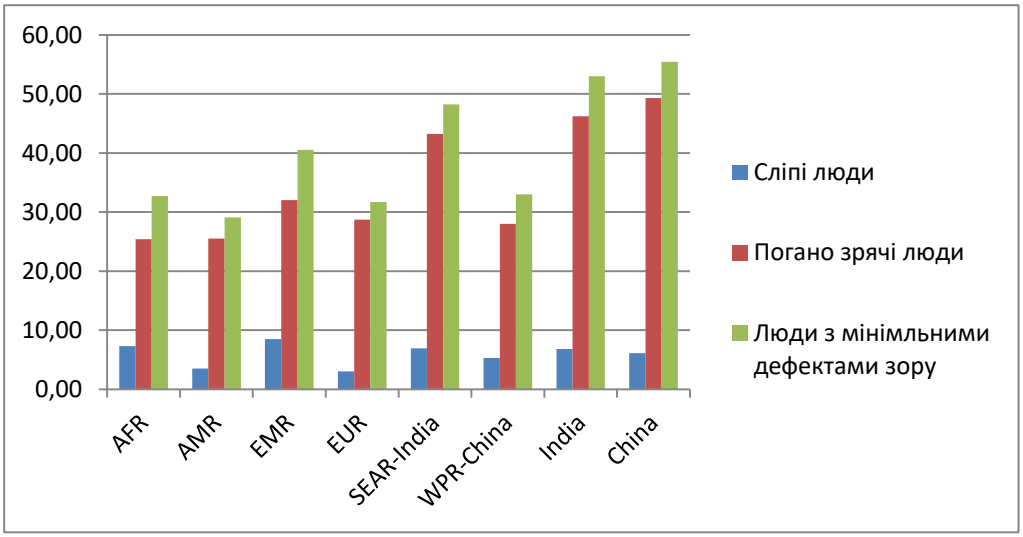

Рисунок 1.6 — Статистика кількості людей із проблемами зору у світових регіонах

Також люди, які ідентифікують себе як глухі, слабочуючі стикаються з численними перешкодами під час доступу до цифрових пристроїв — від комп'ютерів і планшетів до смартфонів і відеоігор. Найпоширеніші перешкоди включають медіа без субтитрів, аудіо без транскриптів, відсутність усного перекладу жестовою мовою та неінклюзивні платформи соціальних мереж. Субтитри або їх відсутність залишаються головним недоліком відомих медіа-організацій. Основні медіа та творці часто завантажують контент без підписів. Або якщо вони є, але тривають лише перші кілька секунд, необхідні для зарахування «перегляду», перш ніж додати текст «натисніть, щоб отримати звук». Під час відеоігор глухим людям може бути дуже важко почути кроки, навколишні шуми та все, що видає звуки. Глухі люди не можуть витрачати свої гроші на придбання таких інструментів, як вібраційні жилети чи навушники, щоб покращити свої умови. Ці недоліки поширюються на такі інструменти для відеоконференцій, як Zoom, технологія живих субтитрів яких зазвичай затримується, важко слідкувати та може спричинити когнітивне перевантаження. На жаль, для спільнот глухих і слабочуючих, перелік перешкод є значним. До них належать низька якість аудіо, відсутність інтерпретації тексту, аудіо та відео, а також обмежені можливості підтримки клієнтів, такі як лінії підтримки по телефону/аудіо. Багато основних платформ соціальних медіа, які приваблюють мільйони користувачів і відіграють важливу роль у поширенні інформації, також приділяють мало уваги спільнотам глухих і слабочуючих. При роботі із різного роду відеохостингами такими, як TikTok та Youtube присутність субтитрів необов'язкова складова для викладення відео, а автоматичні генератори субтитрів на платформах працюють не належним чином. YouTube працює над тим, щоб зробити інструменти субтитрів більш помітними на своїй платформі, але я вважаю, що потрібно підвищити обізнаність про переваги субтитрів і про те, наскільки легко це зробити.

Також існує проблема з функцією автоматичних субтитрів YouTube, яка замінює будь-яке лайливе слово на «[__]». Це викликало критику з боку

глухих і слабочуючих користувачів через той простий факт, що до них ставляться до нас як до дітей, які не можуть переглядати контент із лайливими словами». Приблизно від 2 до 3 з кожної 1000 дітей народжуються з виявленим рівнем втрати слуху в одному або обох вухах. Приблизно 15% дорослих віком від 18 років повідомляють про певні проблеми зі слухом. Це значення і відсотках з роками залишається сталим, але кількість людей із проблемами зору росте у зв'язку із ростом населення світу.

Епілепсія — це тривале (хронічне) захворювання, яке викликає повторювані напади внаслідок ненормальних електричних сигналів, що виробляються пошкодженими клітинами мозку [7].

Спалах неконтрольованої електричної активності в клітинах мозку викликає судомний напад. Судоми можуть включати зміни вашої свідомості, контролю над м'язами (ваші м'язи можуть посмикуватися або смикатися), відчуття, емоції та поведінку. Епілепсію також називають судомним розладом. У всьому світі близько 65 мільйонів людей хворі на епілепсію. Що відбувається, коли у людини стається напад епілепсії клітини вашого мозку надсилають повідомлення та отримують повідомлення з усіх частин вашого тіла. Ці повідомлення передаються за допомогою безперервного електричного імпульсу, який передається від клітини до клітини. Епілепсія порушує цей ритмічний електричний імпульс. Натомість виникають спалахи електричної енергії — як непередбачувана гроза — між клітинами в одній або кількох областях вашого мозку. Цей електричний збій викликає зміни у вашій свідомості (включаючи втрату свідомості), відчуттях, емоціях і рухах м'язів [8].

Люди, які страждають на світлочутливу епілепсію, тобто напади викликаються висококонтрастними візерунками, мерехтливим світлом, стробуючим світлом або миготливим світлом. Вони також можуть мати напади в інший час, але ці ситуації, здається, викликають судоми – переважно тоніко-клонічні.

Отже, людина з епілепсією, яка має доступ до Інтернету, має бути обережною щодо типу вмісту, до якого вона має доступ. Навіть перегляд відео в соціальних мережах може призвести до нападу, якщо вони зіткнуться з вмістом, який може викликати напад.

Судоми у людей можуть бути спровоковані впливом деяких із наведених нижче ситуацій:

— екрани телевізорів або комп'ютерних моніторів через мерехтіння або рухоме зображення;

— певні відеоігри або телевізійні трансляції, що містять швидкі спалахи або чергування різних кольорів;

— певні візерунки, особливо смуги контрастних кольорів.

Люди з такими вадами зазвичай ведуть щоденики тригерів, де записують, дії, які вони виконали під час певного періоду, свій раціон харчування, режим сну і т.д. У Інтернеті величезна кількість небезпеки для таких людей. Більшість потенційно небезпечних web-продуктів навіть не мають необхідного попередження про можливі наслідки освоєння контенту, що може сричинити до найгірших наслідків. Адже навіть банальнее попередження про то, що наприклад у відео на сайті дуже велика частота змін кадрів або велика частота певного мерехтіння, може зберегти комусь життя та здоров’я. Інтернет має бути звичайною буденністю для всіх людей, а не небезпекою для них.

1.4 Стан доступності сайтів у всьому світі

Перш ніж перейти до більш конкретних типів статистики щодо цієї важливої та складної проблеми, давайте подивимося на загальну картину.

— 59,6% людей з інвалідністю проживають у домогосподарствах з доступом до Інтернету;

— 62% дорослих з обмеженими можливостями володіють ноутбуком або настільним комп’ютером;

— 72% дорослих людей з обмеженими можливостями мають смартфон.

Під час дослідження понад мільйона домашніх сторінок у лютому 2021 року було виявлено в середньому 51,4 помилки доступності на сторінку; кількість, яка зменшилася на 15,6% від кількості виявлених помилок у лютому 2020 року (60,9 помилок). Тепер, коли ви можете зрозуміти масштаби ситуації, давайте подивимося на дивовижні цифри, пов’язані з доступністю [9].

Скільки людей потребують модифікації доступності веб-сайту? Незважаючи на те, що мільйони людей живуть з обмеженими можливостями, ці вади можуть виглядати по-різному від людини до людини. Не кожна інвалідність негативно впливає на здатність людини отримати доступ до Інтернету через комп’ютер або смартфон, але багато з інвалідностей, які впливають на людей, створюють бар’єри та труднощі під час доступу та обробки деяких веб-сторінок.

Ось цифри щодо кількості людей з обмеженими можливостями, які роблять функції доступності необхідними під час серфінгу в Інтернеті. У всьому світі принаймні 2,2 мільйона людей мають порушення зору поблизу або вдалину, що може потребувати екранної лупи для веб-сторінок. Понад 466 мільйонів людей у всьому світі мають вади слуху. 5,9% людей глухі або мають серйозні проблеми зі слухом. Приблизно чверть американців з обмеженими можливостями (26%) стверджують, що вони мають високошвидкісний Інтернет вдома, смартфон, настільний або портативний комп’ютер і планшет у порівнянні з 44% тих, хто повідомляє, що не мають інвалідності, 10,8% відсотків людей з обмеженими можливостями мають розлад пізнання з серйозними труднощами з концентрацією, запам’ятовуванням або прийняттям рішень, 8,2% дорслих мають труднощі з підняттям або хапанням, що може вплинути на використання ними миші чи клавіатури, 3,3% населення мають ті чи інші порушення зору (включаючи кольорову сліпоту). Ці люди можуть покладатися на екранну лупу чи зчитувач або потребують більш контрастних

сторінок, які підходять для дальтонізму, 2 з 5 дорослих у віці 65 років і старше мають інвалідність [9].

Очікується, що до 2060 року кількість людей віком 65 років і старше подвоїться, що зробить проблему зростаючою, 75% людей з обмеженими можливостями повідомляють, що користуються Інтернетом щодня. Очікується, що до 2050 року майже 2,5 мільярда людей матимуть певний ступінь втрати слуху, і щонайменше 700 мільйонів потребуватимуть реабілітації слуху. Більше 1 мільярда молодих людей ризикують втратити слух через небезпечну практику прослуховування, якої можна уникнути.

На жаль, багатьом веб-сайтам дуже не вистачає модифікацій доступності, які роблять їхні сторінки доступними для людей з певними обмеженими можливостями.

Це найважливіший аспект, який потрібно вдосконалити кожному власнику веб-сайту, щоб забезпечити легке використання та позитивний досвід користувачів для всіх людей, 97,4% з мільйона найкращих веб-сайтів світу не мають повної доступності. Низькоконтрастний текст, який був нижче порогів WCAG 2 AA, був знайдений на 86,4% домашніх сторінок. Це була проблема доступності, яка найчастіше виявлялася: середня домашня сторінка мала 31 виразний випадок низькоконтрастного тексту, 26% усіх зображень домашньої сторінки (в середньому 10 на сторінці) мали відсутній альтернативний текст. Майже половина зображень, на яких відсутній альтернативний текст, були зображеннями-посиланнями, у результаті чого посилання не мали опису, 45% із 4,4 мільйона виявлених вводів форми не були належним чином позначені (через <label>, aria-label або aria-labelledby). У деяких конкретних сферах онлайн-бізнесу статистика помилок на домашній сторінці гірша за середню, найгіршою є покупка з 75,2 помилками на сторінку. З понад 1 мільйона відібраних сайтів 21,9% сторінок мали 5 або менше виявлених помилок, а 29,9% — 10 або менше, 60% користувачів програми зчитування з екрана вважають, що доступність веб-контенту погіршується.

Домашні сторінки, які використовують загальну систему Google AdSense, мали в середньому на 27 помилок більше, ніж інші сторінки. Лише 21,4% програм веб-доступності мають централізований бюджет. Трохи більше 20% програм доступності взагалі не мають бюджету. 43,9 % програм доступності мають бюджет, розподілений між відповідальними департаментами Хоча на перший погляд деякі з цих цифр можуть здатися зневажливими, багато веб-сайтів почали вживати заходів для сприяння більш іяклюзивним практикам у робочих функціях своїх сайтів [9].

Ось кілька способів, за допомогою яких компанії працюють над створенням кращого майбутнього для веб-доступності. Понад 36% учасників опитування оцінили письмову політику своєї організації та зобов'язання щодо доступності як «Проактивні», що є найвищою оцінкою. Майже 40% учасників поставили собі оцінку «В роботі». Більшість програм доступності віком до 7 років. Середній бізнес працює над доступністю від 2 до 3 років.

Основна причина того, що компанії докладають зусиль для покращення веб-доступності, це включення людей з обмеженими можливостями. Головною причиною, чому багато компаній не мають досконалого протоколу доступності, є час, необхідний для підтримки (65,6%). Організації будь-якого розміру, галузей і віку вибрали «Розвиток програми доступності» серед п'яти головних цілей на 2021 рік (42,6%)

В опитуванні 2021 року респонденти оцінили письмову політику/зобов'язання своєї організації на 39,2% у виконанні та на 36,7% упереджено, 62,8% організацій мали задокументовану процедуру вирішення проблем із доступністю для клієнтів/користувачів, а 62,1% вважали, що скарги щодо доступності вирішувалися швидко.

Понад половина респондентів опитування, які займаються аудитом доступності (57,8%), проводили офіційний аудит доступності протягом останніх шести місяців. Як можна побачити цифри жахливі, але опираючись на оитування респондентів, багато компаній працюють над доступність і ставлять окращення цього параметру, як найпріорітетніші цілі.

2 АНАЛІЗ МЕТОДІВ ПОКРАЩЕННЯ ДОСТУПНОСТІ САЙТІВ

Найкраще враховувати доступність на ранніх стадіях життєвого циклу продукту — це зменшує час і гроші, які ви витратите, щоб зробити свої продукти доступними заднім числом. Доступність кольорів вимагає невеликої попередньої роботи під час вибору палітри кольорів вашого продукту, але забезпечення доступності ваших кольорів принесе дивіденди в майбутньому.Дизайн є, якщо не найважливішим аспектом хорошої доступності, то на рівні з розробкою та кодуванням. У цьому розділі буде розглянуто ключові дії для хорошого рівня доступності на будь-якому сайті.

2.1 Вибір шрифтів та їх використання.

Деякі шрифти легше зрозуміти, ніж інші. Доступний шрифт — це шрифт, який не виключає та не сповільнює швидкість читання будь-якого відвідувача веб-сайту, включно зі сліпотою, втратою зору та розладами читання. Вибір правильного шрифту покращує розбірливість і читабельність вашого сайту для всіх, допомагаючи охопити ширшу аудиторію в Інтернеті. Якщо ви використовуєте недоступний шрифт на своєму веб-сайті, ви наражаєтесь на ризик судового позову. Використання доступних шрифтів, які відповідають Рекомендаціям щодо доступності веб-вмісту (WCAG), має важливе значення для дотримання основних законів про доступність веб-сайтів у США.

Отож, які саме шрифти можна назвати доступними? Не всі шрифти були розроблені з урахуванням веб-доступності. Розмір, колір і контрастність є трьома ключовими факторами, які визначають, чи доступний шрифт.

Щоб відповідати принципам інклюзивного дизайну, важливо вибрати простий, без прикрас і зрозумілий шрифт. Один із найпростіших способів звузити вибір — це знати, яких особливостей у шрифті слід уникати [10].

Недоступні шрифти, як правило, мають одну або декілька з таких характеристик:

— зроблять вміст важчим для читання;

— затруднює розрізнення форм різних літер і символів;

— уповільнює читача;

— затруднює відокремлення однієї літери від іншої за допомогою накладання символів або літер;

— є декоративними або містять непотрібні прикраси Є спеціальні відображувальні шрифти, такі як рукописний стиль, власний або курсивний.

Більшість інформації в Інтернеті – і цінність, яку пропонує ваш бізнес – передається за допомогою тексту. Оскільки величезна кількість людей страждає від втрати зору, ви ризикуєте поставити під загрозу свою репутацію, ігноруючи потреби веб-сайту цієї групи. Ця цифра включає людей, які відчувають проблеми із зором, навіть якщо носять коригуючі лінзи або контактні лінзи. І це число продовжуватиме зростати, оскільки населення старіє та відчуває більше проблем із зором, пов’язаних із віком і захворюваннями [10].

Не лише людям із вадами зору важко читати певні шрифти. Люди з труднощами в навчанні, як-от дислексія, також можуть бути чутливими до певних шрифтів. Ваш вибір типу шрифту також може значно вплинути на рівень читабельності. Варто зазначити, що дислексією страждає до 20% населення.

Забезпечивши доступність шрифтів вашого веб-сайту, ви зможете краще охопити цю значну частину населення та захистити репутацію свого бренду як організації, яка серйозно ставиться до інклюзивності.

Використання недоступного шрифту може призвести до серйозних юридичних і фінансових санкцій, таких як штрафи, судові позови та інші примусові заходи.

Вибираючи доступний шрифт, пам’ятайте, що доступність і цифрове включення не починаються і не закінчуються на ваших веб-сторінках. Розгляньте всі способи, якими ви використовуєте письмові повідомлення для зв’язку зі своєю цільовою аудиторією, наприклад електронні листи, цільові сторінки, PDF-файли, відео та зображення.

Усі ці формати вмісту мають бути написані однаковим шрифтом. Якщо це неможливо, слід використовувати якомога менше типів шрифтів.

Було б гарною ідеєю вказати вибір доступного шрифту в інструкціях щодо бренду та переконатися, що всі зацікавлені сторони веб-сайту знають про різні рівні веб-доступності різних шрифтів.

Щоб досягти веб-доступності та мінімізувати плутанину для відвідувачів вашого веб-сайту, ви повинні використовувати якомога менше шрифтів на своєму веб-сайті. Але це правило не допомагає вибрати найкращий шрифт для веб-доступності. Не завжди очевидно, які шрифти доступні.

На щастя, стандарти доступності, такі як WCAG, допомагають пролити світло на те, які шрифти забезпечують найбільш інклюзивний веб-сайт для всіх користувачів.

Хороша новина полягає в тому, що вам не потрібно інвестувати в спеціалізований спеціальний шрифт, щоб зробити вміст вашого веб-сайту доступним. Багато стандартних і широко доступних шрифтів мають високу оцінку доступності в Інтернеті. Найдоступнішими шрифтами є Tahoma, Calibri, Helvetica, Arial, Verdana та Times New Roman.

Шрифти Slab Serif, зокрема Arvo, Museo Slab і Rockwell, також вважаються доступними. Ці типи шрифтів переважно використовуються в заголовках, а не в основному тексті.

Можливо, ви чули, що шрифти без зарубок більш доступні для читання з екрана. Однак, оскільки дослідження не є переконливими щодо того, які гарнітури із засічками чи без засічок є більш читабельними, рішення залежить від вас. Для найкращого результату рекомендується вибирати загальні шрифти або сімейства шрифтів, які мають сильні та унікальні символи.

Вам слід вибрати більш поширені шрифти замість менш популярних, щоб збільшити ймовірність того, що пристрій відвідувача вашого веб-сайту зможе правильно відобразити його.

Уникайте шрифтів із «самозваними формами літер», які створені так, щоб бути дуже схожими на інші форми літер як частину їхнього візуального стилю, наприклад, верхній регістр, цифра 1 і малі L. Для більшого розуміння чому це так важливо і без деталей розглянемо приклади поганих і хороших шрифтів. На рисунку 2.1 зліва зобржені шрифт из хорошою читабельністю, а з правої частини шрифти, які не варто використовувати при роботі над доступністю.

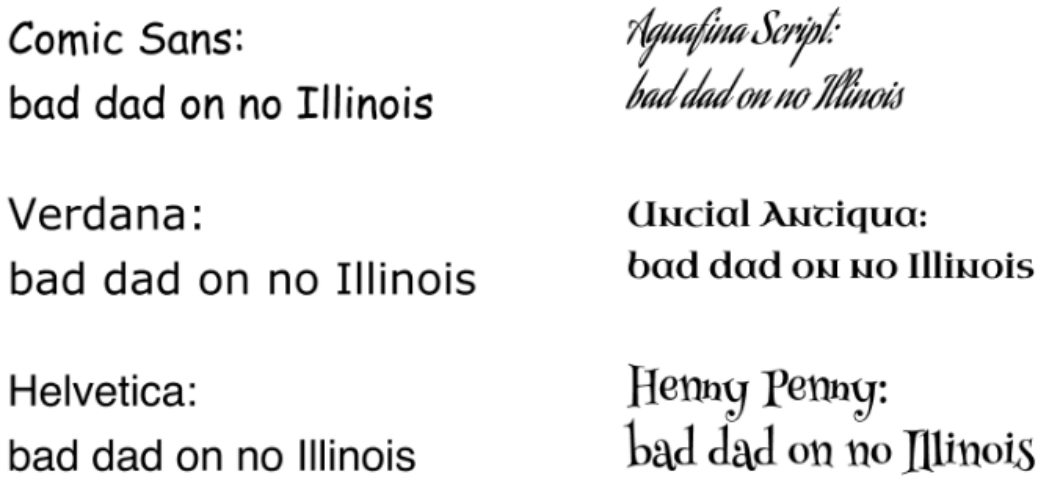

Рисунок 2.1 — Приклади читабельних та не читабельних шрифтів

Навіть якщо ви вибрали інклюзивний шрифт, текст вашого веб-сайту все одно може бути недоступним. Деяким відвідувачам вашого веб-сайту знадобиться більший або менший розмір шрифту для оптимального читання. Використання шрифту неправильного розміру також може призвести до того, що ваш вміст буде невідповідним стандартам веб-доступності.

Не дозволяйте відвідувачам вашого веб-сайту збільшувати або зменшувати розмір тексту, не збільшуючи все інше на сторінці.

Забезпечення розміру тексту дозволяє належним чином відображати ваш вміст на всіх пристроях, включаючи мобільні телефони, планшети та програми зчитування з екрана.

Що стосується розмірів шрифту, існує чотири різні одиниці вимірювання:

— ems (em) ця величина визначає розмір шрифту відносно розміру батьківського елемента на сторінці HTML;

— rems (root ems) ця величина визначається відносно розміру шрифту кореневого елемента в HTML;

— пікселі (px), не порівнюються з базовим елементом, вони є абсолютною одиницею вимірювання;

— points (pt) — ще одна абсолютна одиниця вимірювання.

Рекомендується визначати розмір шрифту у відносних одиницях, таких як відсотки, rem або em, замість абсолютних одиниць вимірювання, таких як пікселі або точки. У деяких браузерах неможливо збільшити текст у пікселях окремо від решти веб-сторінки. З іншого боку, використання відносного шрифту дозволяє відповідним чином змінювати розмір тексту на багатьох пристроях і платформах.

Але коли справа доходить до розміру, законодавство про доступність, як-от ADA, і стандарти, як-от WCAG, не вказують офіційний мінімальний розмір шрифту для веб-тексту. Це не означає, що доступний будь-який розмір шрифту — якщо текст занадто великий або маленький, його може бути надто складно читати. Тому найкраще дозволити відвідувачам вашого веб-сайту самостійно вибирати оптимальний розмір шрифту за допомогою збільшення.

Стандарти відповідності WCAG передбачають такі вимоги щодо збільшення для доступності тексту: «За винятком підписів і зображень тексту, розмір тексту можна змінювати без допоміжних технологій до 200 відсотків без втрати вмісту чи функціональності».

2.2 Підбір кольорової палітри та коректне використання кольорів

Коли справа доходить до дизайну веб-сайту, колір часто використовується для надання значення. Такий вибір дизайну насправді може зробити ваш вміст недоступним.

Щоб усім було простіше точно бачити та сприймати ваш вміст, WCAG рекомендує не використовувати колір шрифту як єдиний візуальний засіб

передачі інформації. Це включає в себе використання відмінностей кольорів для спонукання до відповіді користувача або для інтерпретації візуального елемента на вашій веб-сторінці, наприклад кольорових, але не підкреслених гіперпосилань у реченні або елементів у списку, де деякі представлені кольоровим текстом, щоб показати різницю.

Якщо вам потрібно використовувати колір шрифту для передачі інформації, обов'язково включіть альтернативні візуальні індикатори, щоб допомогти людям із слабким сприйняттям кольору точно інтерпретувати її. Це може включати такі дії, як підкреслення, виділення жирним шрифтом, виділення курсивом або використання інших помітних ознак, для яких не потрібне повне кольорове бачення, щоб візуально відрізняти його від навколишнього тексту.

Контраст — варто зауважити, що закони та рекомендації щодо доступності прямо не забороняють використання будь-яких конкретних кольорів або комбінацій кольорів для веб-тексту. Натомість колірний контраст є мірою, яка використовується для вимірювання відповідності доступності.

Кольоровий контраст — це колірний контраст між текстом і фоном, на якому він відображається. Використання достатнього колірного контрасту для вашого тексту полегшить усім — і особливо людям із слабким зором – чітке бачення вашого веб-тексту. Занадто великий (або занадто низький) контраст може спричинити проблеми. Подумайте про те, щоб спробувати прочитати кольоровий текст із низьким контрастом, наприклад сірий текст на білому фоні, у сонячний день на вулиці [11].

Чорний текст на білому фоні є стандартним для веб-вмісту, але багато веб-сайтів відхиляються від цього з міркувань брендингу та стилістики – і не всі ці комбінації будуть доступними. Корисно те, що в інструкціях WCAG чітко зазначено рівні контрастності, необхідні для того, щоб текст вважався доступним:

Заголовки: Коефіцієнт контрастності для великомасштабного тексту має бути не менше 3 до 1.

Основний текст: для тексту, крім заголовків, слід підтримувати коефіцієнт контрастності принаймні у відношенні 4,5 до 1. Тому, розробляючи такі речі, як кнопки, картки чи навігаційні елементи, обов’язково перевірте коефіцієнт контрастності ваших комбінацій кольорів. На рисунку 2.2 зображено два приклади роботи із контрастом.

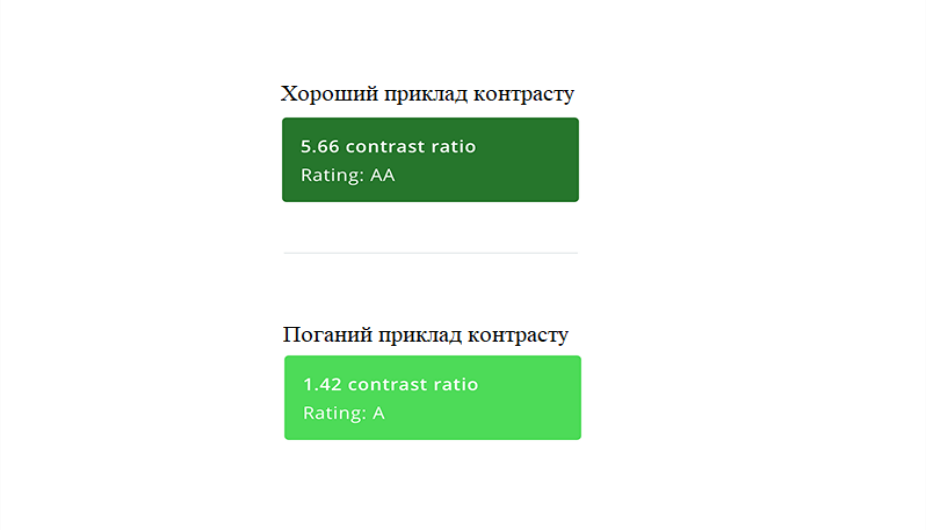

Рисунок 2.2 — Приклади роботи із контрастом

Логотип: Текст, який є частиною логотипу чи бренду, не має вимог щодо контрасту.

Випадковий текст: Текст, який є частиною неактивного інтерфейсу користувача, є чистою прикрасою, невидимим або частиною зображення та не передає значущої інформації, не вимагає контрасту.

Якщо ви не є фахівцем у веб-доступності, скористайтеся професійним інструментом перевірки контрастності кольорів, як-от Color Contrast Checker від Siteimprove, — це найпростіший і найнадійніший спосіб перевірити доступність коефіцієнтів контрастності кольорів вашого веб-тексту.

Також не варто покладатися лише на колір. Ви також можете забезпечити доступність, переконавшись, що ви не покладаєтеся на колір для передачі важливої системної інформації. Отже, для таких речей, як стани помилок, стани успіху або системні попередження, обов’язково додайте

повідомлення чи іконографію, які чітко вказують на те, що відбувається. Приклади зображені на рисунку 2.3.

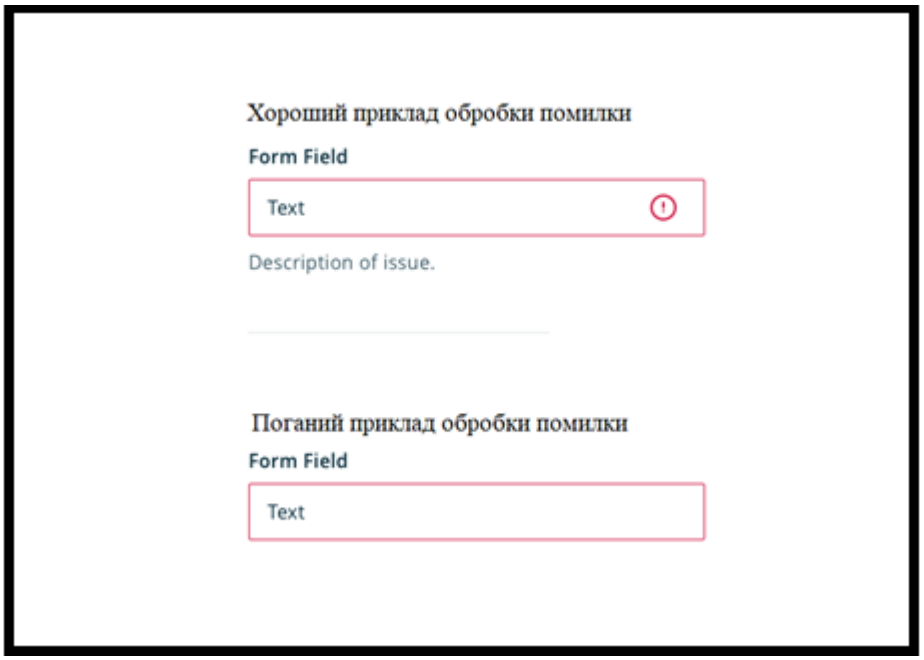

Рисунок 2.3 — Приклади обробок помилок для користувача

Крім того, під час відображення таких речей, як графіки чи діаграми, надання користувачам можливості додавати текстури чи візерунки гарантує, що дальтоніки зможуть їх розрізняти, не турбуючись про те, що колір впливає на сприйняття даних. Trello чудово справляється з цим завдяки своєму режиму для дальтоніків. Для більшого розуміння прклади таких текстур зображені на рисунку 2.4.

Рисунок 2.4 — Приклад текстур для діаграм

Стани фокусування допомагають людям переміщатися по вашому сайту за допомогою клавіатури, надаючи їм візуальний індикатор навколо елементів. Вони корисні для людей з вадами зору, людей з обмеженими руховими можливостями та людей, які просто люблять орієнтуватися за допомогою клавіатури.

Усі браузери мають колір стану фокусу за замовчуванням, але якщо ви плануєте замінити його у своєму продукті, дуже важливо переконатися, що ви забезпечуєте достатній колірний контраст. Це гарантує, що люди з вадами зору або недоліками кольору можуть переміщатися за допомогою станів фокусування [11].

Насамкінець, найважливішим аспектом створення доступної системи кольорів є надання вашій команді можливості посилатися на неї за потреби, щоб усі зрозуміли правильне використання. Це не тільки зменшує плутанину та відтік, але й гарантує, що доступність завжди буде пріоритетом для вашої команди. З мого досвіду, чітке визначення рейтингу доступності певної комбінації кольорів у наборі інтерфейсу користувача чи системі дизайну є найефективнішим, особливо під час спілкування в команді за допомогою інструменту (InVision Craft або InVision DSM). Ось приклад того, як задокументувати комбінації кольорів фону для тексту та рейтинг доступності кожної комбінації. Приклад такої палітри зображений на рисунку 2.5.

Рисунок 2.5 — Приклад сформованої кольорової палітри для сайтів

2.3 Технічні методи покращення доступності

Перше, що слід враховувати, вивчаючи, як зробити свій веб-сайт доступним, — це знати, які люди використовують ваш веб-сайт, як вони це роблять і як вони цього хочуть. Розмірковуючи про доступність, пов’язану з обмеженими можливостями, ви можете поглянути на те, як програмне та апаратне забезпечення допомагають людям користуватися комп’ютером.

2.3.1 Основи, які слід враховувати

Люди з поганим зором можуть збільшити сторінку. Але слід пам'ятати, що не всі порушення зору працюють так. Наприклад, люди з тунельним зором вважають за краще зменшувати зображення через менше поле зору. Пам’ятайте, що люди, які все ще бачать, навіть ледве бачать, завжди хочуть використовувати свій зір, коли це можливо.

Сліпі люди або люди з дуже поганим зором можуть використовувати ваш веб-сайт із програмами зчитування з екрана або дисплеями Брайля.

Не всі користуються мишкою. Це або через уподобання (розробники про це чудово знають), або через фізичні обмеження. Замість миші люди можуть використовувати лише клавіатуру, програмне забезпечення для розпізнавання голосу або апаратне забезпечення, як-от перемикач.

У покращенні доступності сайтів відіграє велику роль HTML розмітка. HTML містить дуже прості та легкі у використанні теги, які надають семантичну цінність веб-сторінці. Потрібно лише знати про них і правильно ними користуватися. Деякі загальні практити:

Пам'ятайте про структуру заголовк. <h1> має бути першим, і має бути лише один. Під <h1> має бути <h2>, під <h2> — <h3> і так далі.

Завжди надавайте альтернативний текст, який є коротким і описовим для зображень. Однак, якщо зображення лише для декоративних цілей, залиште поле альтернативного тексту порожнім, але збережіть атрибут alt.

Створення нового абзацу за допомогою <p> семантично не те саме, що додавання нового рядка за допомогою
. Візуально результат може виглядати однаково, але інтерпретується по-різному.

Використовуйте теги <ol> і <ul> для списків. Ніколи не робіть їх вручну.

Додаючи мітки до <input>, пам’ятайте про атрибут for, щоб анотувати, який <input> він описує. Крім того, пам’ятайте, що placeholder не призначені для використання замість label.

Спробуйте дотримуватися звичайних input для форм. Хоча спеціальні поля зі списком або засоби вибору дати можуть виглядати привабливо, це не означає, що їх легко використовувати з інструментами доступності. Введення за замовчуванням зроблено з урахуванням доступності.

Однак іноді ми просто хочемо зробити щось із багатьох і <spans>, щоб це виглядало добре. Якщо це так, ви можете використовувати один із атрибутів ARIA, щоб додати семантичні елементи. Наприклад, якщо ваш фактично є прапорцем, ви можете встановити *role*=”checkbox” разом із aria-checked.

Дуже важливо пам’ятати те, що інструменти доступності не знають CSS або JS. Створюючи веб-сторінку з програмами зчитування з екрана, ви повинні пам’ятати, що ці інструменти не аналізують стилі CSS і сценарії JavaScript. Ось чому, якщо ви хочете підкреслити щось, незважаючи на додавання гарних стилів, ви також повинні пам’ятати додати для цього відповідний тег HTML.

2.3.2 Використання сценаріїв

Якщо ви додали подію onClick до якогось елемента, програми зчитування з екрана не розпізнають це, доки ви не позначите її відповідними атрибутами, як описано раніше. Також пам'ятайте про порядок елементів. Якщо ви змінили його в CSS, наприклад, порядок елементів у Flex, програми зчитування з екрана не розпізнають його.

Іншою проблемою, з якою можуть зіткнутися користувачі розпізнавання голосу, є погано описані посилання. Уявіть, що у вас немає миші, і ви

розмовляєте лише з комп’ютером. Ви хочете клікнути на посилання, але кожне з них називається «натисніть тут» або «(посилання)». Як би ви натиснули той, який хочете? В іншому випадку кожна кнопка описується піктограмою. Програма зчитування з екрана не читає значення піктограми. Крім того, як би ви назвали посилання з інструментом розпізнавання мовлення?

Тому посилання повинні мати більше описових текстів. Якщо ви хочете використовувати піктограми, вам слід додати додаткові мітки для програм зчитування з екрана. Якщо повернутись до пункту із збільшенням та зменшенням сторінки. Це широко поширена можлиівсть якою користуються багато людей. Щоб переконатися, що все працює правильно, вам потрібно зробити ваші веб-сторінки повністю адаптивними. Так, швидкість responsebility — це більше, ніж забезпечення роботи сторінки на мобільних пристроях. Це також зміна розміру вікна та збільшення масштабу. Навіть якщо ви не орієнтуєтеся на мобільні пристрої, пам’ятайте про те, щоб зберегти якийсь адаптивний макет. Одна з найважливіших речей — не змушувати користувачів прокручувати текст горизонтально, щоб прочитати текст. Дуже легко втратити контекст того, що ви читаєте, коли вам потрібно прокрутити праворуч, а потім ліворуч під час переходу до наступного рядка. З іншого боку, вертикальне прокручування ні для кого не є проблемою [12].

Навіть якщо ви не знайомі з інструментами доступності та не маєте жодних знань про стандарти, ви все одно можете виконати деякі автоматизовані тести. Ось деякі з них, які я можу вам порекомендувати:

Збільшуйте, навіть до 500%, зменшуйте та змінюйте розмір вікна. Загалом, пограйте з чуйністю.

У браузерах Firefox і інструментах розробника ви можете знайти вкладку «Доступність». Там ви побачите структуру сторінки так, як її бачать програми зчитування з екрана, а також деякі основні помилки та попередження.

Якщо ви не використовуєте Firefox або вам потрібен більш зручний інструмент, я рекомендую Інструмент оцінки веб-доступності (WAVE). Це розширення для браузера, яке працює з Chrome і Firefox. Він може відображати всі помилки саме там, де вони є на сторінці, і підсумовувати те, що ви зробили добре! Він також має зручну функцію для вимкнення всіх стилів CSS, що дає змогу побачити, як ваші стилі впливають на читабельність сторінки.

2.3.3 Додаткові способи перевірки

Щоб перевірити речі іншим способом, вам слід завантажити розширення браузера Web Developer для Firefox, Chrome і Opera. Це дозволяє вносити багато змін у те, як браузер відображає сторінку. З точки зору доступності, я вважаю корисним приховати всі зображення та замість них показати їхній альтернативний текст. Зробивши це, ви зможете побачити, чи все ще знаєте, про що йдеться в нашому вмісті. Він також пропонує багато інших корисних функцій для веб-розробників, тому його варто перевірити.

Ви можете просто запустити програму зчитування з екрана та перевірити свою сторінку за допомогою неї. Для цього є вбудовані програми в macOS, Windows, iOS і Android. Не бійтеся спробувати. Ви все одно бачитимете свій екран, і його завжди можна швидко вимкнути.

Намагайтеся використовувати свою сторінку лише за допомогою клавіатури. Використовуйте «Tab», щоб перейти до наступного введення, посилання або кнопки, і «Shift+Tab», щоб перейти до попереднього. Ви використовуватимете кнопку «Enter» замість клацання мишею та стрілки замість колеса миші.

Зрозуміло, що ви не зможете повністю усвідомити накільки ваш додаток зручний для всіх, тому для людини із здоровим зором не можливо нормально показати їх роботу очима людей із поганим зором, тому тестування є важливим перевірити усе у найскладніших умов із максимальним зумом, мінімальною видимістю, спробувати освоїти контент сайту без миші лише з клавіатурою або взагалі вимкнувши свій екран чи монітор.

2.4 Огляд сайтів та аналіз типових помилок при роботі із доступністю

При аналізі сайтів будуть розглянуті лише домашні сторінки сайтів для економії часу і огляду максимально великої кількості сайтів. При огляді буде звертатись увага як і на візуальні помилки так і н технічні. Візуальні помилки будуть підтверджуватися стандартами Web Content Accessibility Guidelines, а технічні будуть визначатися задопомогою інструментом у браузері під назвою «Lighthouse».

2.4.1 Google Lighthouse

Google Lighthouse — це безкоштовний інструмент із відкритим вихідним кодом, який може допомогти вам покращити швидкість, продуктивність і загальну взаємодію з веб-сайтом. Створення звітів Lighthouse полегшує покращення якості ваших веб-сторінок [13]. Lighthouse не лише для розробників. Інструмент підходить для будь-якого власника сайту, який хоче дізнатися більше про ефективність свого веб-сайту та конкретні кроки для його оптимізації. Перейдемо до того як Lighthouse перевіряє сторінки, аудит Lighthouse зосереджується на основних веб-показниках Google. Якщо ви не знайомі, це основні показники, які Google використовує для вимірювання швидкості веб-сторінок і загальної взаємодії з користувачем. Вони складаються з малювання найбільшого вмісту (LCP), затримки першого введення (FID) і сукупного зсуву макета (CLS) [13]. Іншими словами, використання Lighthouse допомагає вам бачити свій веб-сайт так само, як Google. Ви можете використовувати корисну інформацію, яку він надає, щоб оптимізувати свої сторінки для кращого рейтингу в пошукових системах.

Звіти Lighthouse складаються з п’яти категорій:

— продуктивність
— доступність
— SEO
— хороші практики

— прогресиивний web-додаток

Приклад результату, який показує інструмент зображено на рисунку 2.6.

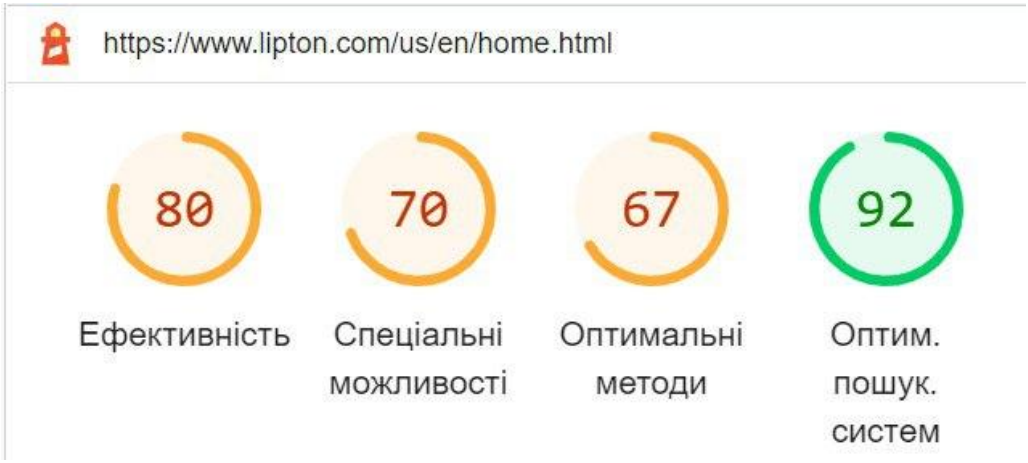

Рисунок 2.6 — Приклад результату роботи ligthouse

Але нас цікавлять тільки показнаки достуності. Далі варто розповісти, які саме критерії вливають на оцінку інструменту. Оцінка Lighthouse Accessibility — це середньозважене значення всіх перевірок доступності.

Кожен аудит доступності може проходити або не проходити. На відміну від аудиту ефективності, сторінка не отримує бали за часткове проходження аудиту доступності. Наприклад, якщо деякі кнопки на сторінці мають доступні назви, а інші ні, сторінка отримує 0 для кнопок, які не мають перевірки доступних імен.

У наведеній нижче таблиці 2.1 показано деякі із зважувань для кожного аудиту доступності. Більш зважені аудити мають більший вплив на ваш бал.

Таблиця 2.1 — Приклади аудитів для інструменту Lighthouse

Аудит	Вага на оцінку
Атрибути [aria-*] відповідають своїм ролям	10
Кнопки, посилання та елементи меню мають доступні імена	3
Аудит	Вага на оцінку
[aria-hidden="true"] немає в <body> документа	10

Документ має елемент <title>	3
Елементи <frame> або <iframe> мають назву	3
Елемент <html> має дійсне значення для атрибута [lang].	3

2.4.2 Огляд популярних сайтів

У пошуках сайтів для огляду було переглянуто багато варіантів, але більшість з них були сайти маловідомих продуктів, але я натикнувся на сайт компанії ‘Lipton’. Компанія достатньо масштабна та продає свою продукцію по всьому світу, включаючи країни, де наявність доступності на сайтах затверджена на законодавчому рівні. На перший погляд сайт такої величезної компанії мав би бути максимально достуним без найменших погрішностей і з однієї сторони, це насравді так і є, тому, що з технічної сторони усе чудово. Видно, що розробники постарались і зробили великий обсяг роботи, тому що все на хорошому рівні, сайт за допомогою скрін-рідера читається чудово. Карта сайту достаньо проста і є стандартною. Також на сайті є окрема сторінка із описом роботи, яка була виконана для підвищення рівня доступності. Але у мене вийшло знайти деякі проблеми та помилки на сайті зі сторони дизайну.

Перейшовши на домашню сторінку сайту користувач одразу бачить меню та першу секцію сайту де знаходиться баннер. При першому огляді відразу помітно, що текст посилань у меню має хороший шрифт і хороший рівень контрасту у двох варіаціях, при стабільному меню і при скролі сторінки.

Але є два недоліки, а саме відстань між літерами посилань, він мінімальний та низька роздільна здатність логотипу. Я вважаю, що це не дуже хороша практика, тому що літери при невеликому віддалені сторінки або на великих моніторах зливаються одна з одною, а логотип взагалі буде суцільним жовтим кольором і погано читабельний навіть у звичайних умовах. Банер та меню показаны на рисунку 2.6. Також для людей навіть з мінімальними порушеннями зору, літери одразу будуть зливатися і цей контент стане

важким для освоєння. Для хорошого зорового порівняння на рисунку 2.7 зображено посилання із відстанню віж літерами, як на сайті на даний момент та після невеликого збільшення. Різниця очевидна, хоча сам шрифт не змінився взагалі, такий жерозмір колір та жирність.

Наступне, що помітно відразу, при бажані виділити текст у банері неможливо через те, що увесь баннер це суцільна картинка. Це дуже велика помилка, адже як було згадано раніше для доступного сайту дуже важливо бути максимально «еластичним» та підлаштовуватись під зміни масштабу і монітори девайсів усіх розмірів.

Рисунок 2.6 — Меню та баннер сайту

Рисунок 2.7 — Посилання із різними відстанями між літерами

Картинка не має таких властивостей і при збільшені не змінюється взагалі тому на мілкий текст важко прочитати майже наусіх моніторах, людям, які користуються скрін-рідерами майже не можливо. На рисунку 2.8

зображений баннер у звичайному масштабі та уже збільшені до 150%. Можна помітити, що елемети меню відреагували на збільшення сторінки чудово, а от баннер не змінився взагалі, це є великою помилкою та поганою практикою.

Прогорнувши сторінку вниз наступною помилкою можна відмітити проблеми із котрастом у контенті. На рисунку 2.8 можна побачити блоки у яких явні проблеми із контрастом, щей на дожачу знову це суцільні картинки, а не окрмий текст та фон.

Рисунок 2.8 — Контент із поганими рівнями контрасту

Навіть не професійним оком помітно, те що білий текст може бути розташований на фоні із світлими тонами, а то й взагалі білому фоні. Також підзаголовок із картинкою майже одного кольору із картинкою на фоні, до того ж маючи стилі прозорості. Тобто можна сказати, що у деяких місцях контент має ввідношення контрасту 1 до 1, у той час коли норма 3 до 1, це не припустимо. Проаналізувавши сайт можна інструмент показав 70% у результаті.

У сервісу також вийшло побачити декілька проколів у коді сайту. Перша і найпоширеніша помилка у тому, що [aria-*] атрибути не відповідають своїм ролям та не мають своїх дійсних значень. У цьому випадку причною таких помилок є використання бібліотеки «slick-slider» для реалізації слайдерів на сайтах, ця бібліотека дозволяє зручно і швидко реалізовувати такі елементи, але контралювати атрибути, які присвоює сама бібліотека часто може бути

проблематично. Також на сайті теги <input> та писилання не мають зрозумілих назв для зчитування їх програмами скрін-рідерами.

Наступний обраний сайт для аналізу став сайт Інтернет-магазину «Rozetka». Це найпопулярніший Інтернет-магазин в Україні, тому перевірка цього сайту буде чудовим прикладом. Оглянувши домашню сторінку сайту можна відразу виділити мілкі проколи у рівні контрасту у евних місцях, для прикладу візьмемо кнопку для активації пошуку на сайті забражену на рисунку 2.10.

Рисунок 2.10 — Поле введеня для пошуку на сайті

Фон кнопки зелений, а саме #00a046 у кодовому представлені, тоді колір текст всередині кнопки білий із кодом #ffffff. Для того, щоб довести це математично і відповідно до стандартів. Для цього нам необхідно порахувати рівень яскравості кольольорів, їхнє відношення і є рівнем контрасту. Формула обчислення яскравості дорівнює:

$$L = 0.2126 * Rg + 0.7152 * Gg + 0.0722 * Bg \quad (2.1)$$

У свою чергу R_g вираховується за формулою:

$$Rg = R/3294 \quad (2.2)$$

І якшо R більше за 10 то фурмула змінюється на:

$$Rg = \left(\frac{R}{269} + 0.0513\right) * 2.4 \quad (2.3)$$

Значення G_g, B_g обраховуються аналогічно до R_g.

Значення білого кольору дорівнє 1, а чорного 21, отже потрібно порахувати рівень кольору #00a046 у представлені RGB колір дорівнює R = 0, G = 160, B = 70. Підставимо ці значення у формули:

$$Rg = 0/3294 \quad (2.4),$$

$$\mathrm{Gg} = \left(\frac{160}{269} + 0.0513\right) * 2.4 \quad (2.5),$$

$$\mathrm{Bg} = \left(\frac{70}{269} + 0.0513\right) * 2.4 \quad (2.6)$$

У результаті R = 0, G = 1.55062929368, B = 0.74765531598. Підставимо отримані значення у формулу яскравості.

$$\mathrm{L} = 0.2126 * 0 + 0.7152 * 1.55062929368 + 0.0722 * 0.74765531598 \quad (2.7)$$

Після обчислень значення яскравості дорівнює 1.16299078465. Щоб визначити рівнь контрасту необхідно скористатись формулою:

$$\mathrm{L} = (\mathrm{L1} + 0.05)/(\mathrm{L2} * 0.05) \quad (2.8)$$

де L_1 — більше значення яскравості

L_2 — менше значення яскравості,

Підставиимо значення у формулу:

$$\mathrm{L} = (1.16299078465 + 0.05)/(1 + 0.05) \quad (2.9)$$

Рівень контрасту при таких значенях дорівнює 1.15522931871, але для звичайного мілкого тексту рівень контрасту має бути 4.5. Настуним кроком

перевіримо технічну роботу над доступністю цього сайту. Відсоток доступності на сайті «Rozetka» складає 78%.

Із помилок у коді можна виділити недоступні для зчитування назви у кнопках та посиланнях. Також присутні атрибути у тезі <meta>, такі як [user-scalable="no"] .

Наступним у огляді буде сайт порталу із новинами «Українська правда». На цьому сайті також траплються проблеми із рівнем контрасту.На сайті присутній текст із кольором #888888 на білому фоні із кодом #ffffff, при таких кольорах рівень контрасту становить 1 до 3.54. Перевіримо технічну роботу над доступністю цього сайту. Сервіс оцінив достуність сайту на 79%.

Такий результат обмумовлений картинками без тегу [alt] без цього тегу картинки не представляють ніякої цінності для скрін-редерів та просто ігноруються. Також у тегів <iframe> відсутній атрибут [title], який має містити короткий опис вмісту підвантаженого контенту, багато посилань не мають описового тексту. Останнюю, але великою помилкою із сторонни семантичного коду є не послідовний порядок використання заголовків.

Наступним буде сайт прогнозу погоди «Sinoptik» . На сайті зустрічаються проблеми ізконтрастом кольорів, присутній колір #9d9d9d у шрифтах на білому фоні.

Ці кольори мають відношення 1 до 2.71. При перевірці коду на доступність отримано результат 73%.

Такий результат обумовлений відсутнісю [alt] атрибуту у тегах <img>, відсутністю атрибуту [lang] у тезі <html>, який описує основну мову сайту.

Тепер випишемо результати усіх сайтів у таблицю 2.2.

Таблиця 2.2 — результати технічного аналізу доступності оглянутих сайтів.

Назва сайту	Відсоток доступності	Прогресивний web-додаток

Закінчення таблиці 2.2.

Lipton	70%	Ні
Rozetka	78%	Так
Українська правда	79%	Ні
Sinoptik	73%	Ні

Якщо підсумувати ці результати, то можна побачити, що в середньому на сайтах відсоток доступності дорівнює приблизно 70% . На сайтах зустрічаються типові помилки у семантиці верстки та недостатня кількість атрибутів для скрін-рідерів, так і їх не точні значення. У дизайнах часто зустрічаються помилки у виборі кольорової палітри та технічнх рішеннях задля зручнішої та швидшої роботи.

3 РОЗРОБКА WEB-ДОДАТКУ

3.1 Розробка клієнтської частини додатку

Для початку потрібно створити проект ті ініціалізувати у проект всі необхідні інструменти. Фреймворк Angular дозволяє ініціалізувати проект за допомогою npm, прописавши дві команди. Порядок команд наведено у лістингу 3.1.

Лістинг 3.1 — Команди для ініціалізації Angular проекту.

```
npm install -g @angular/cli
ng new my-app
cd my-app
ng serve –open
```

Перша команда встановлює інтерфейс командного рядка екосистеми Angular. Команда ng new сворює директорію з проектом та усіма потрібними інструментами Angular. Після чого залишається лише обрати директорію із проектом та запустити проект на локальному сервері. Стандартна та початкова структура для Angular проекту зображена на рисунку 3.1.

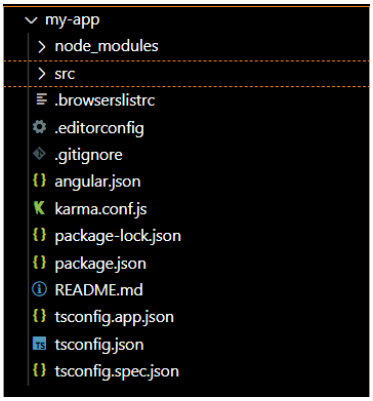

Рисунок 3.1 — Початкова структура Angular проекту

3.1.1 Картка товару

Оскільки в додатку потрібно реалізувати функціонал Інтернет-магазину, варто почати з картки товару. Для початку створимо html структуру картки.

Для створення потрібних файлів компонента скористуємось Angular CLI. Комада ng generate component <name> створює чотири необхідні для розробки компонента файли, а саме для hml верстки, для написання лоігки, для створення unit тестів та для стилів.

При створені html каркасу варто пам’ятати про семантику та правильну структуру. Тому важливо, щоб картка товару була огорнута у тег <article>, цей тег призначений для повністю незалежних елементів типу статі, запису блогу. Також для елементів, які повторно використовуватимуться. Оскільки картка товару не втратить, а ні сенсу, ні у своїх властивостях після зміни розташування тег <article> підходить ідеально. Лістинг коду із html шаблоном компонент наведено у лістингу 3.2.

Лістинг 3.2 — Шаблон html для CardComponent

```
<article class="products__item product-card product-card--with-rating" >
  <div class="product-card__main">
    <div class="product-card__img">
      <img [src]="card.images[0]" alt="product" />
    </div>
    <div class="product-card__info">
      <div class="product-card__wr">
        <div class="product-card__rating">
          <span>
            {{ card.rating }}
          </span>
          <img src="../../../assets/img/icons/star.svg" alt="rating-star" />
        </div>
        <div tabindex="0" class="product-card__price">${{ card.price }}</div>
      </div>
      <a tabindex="0" href="/product" aria-label="product name"
        class="product-card__description"
```

```
      >
        {{ card.title }}
      </a>
    </div>
  </div>
  <div class="product-card__btns">
    <button  tabindex="0" aria-label="add to wishlist" class="product-card__btn
btn">
      <img
        src="../../assets/img/icons/heart.svg"
        alt="heart"
      />
      <span>WISHLIST</span>
    </button>
    <button  tabindex="0"
      aria-label="add to cart"
      class="product-card__btn btn btn--violet"
    >
      <img
        src="../../../assets/img/icons/bag.svg"
        alt="bag"
      />
      <span>
        ADD TO CART
      </span>
    </button>
  </div>
 </article>
```

Оглянемо поля, які буде приймеати компонент картки товару:

— id — унікальний ідентифікатор, який генерується окремо для кожного продукту;

— images — массив із фотографіями продукту;

— title — назва продукту;

— rating — рейтнг продукту серед голосування покупців;

— price — ціна продукту;

— brand — бренд продукту;

— category — категорія продукту;

— isCart — належність продукту до корзини корстувача;

— isWished — належність продукту до списку бажаних товарів користувача.

Скріншот готового компонента із статичними даними зображено на рисунку 3.2.

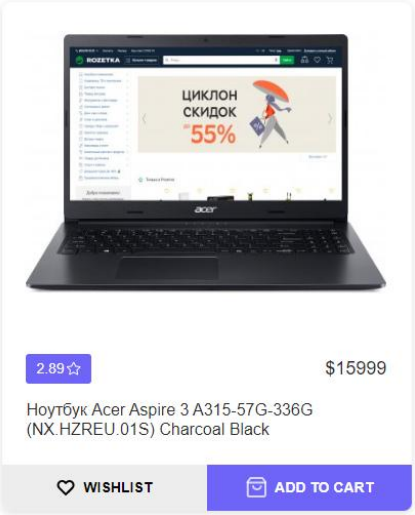

Рисунок 3.2 — Компонент картки із статичнми даннми

Компонента отримує дані, які притаманні сутності продукту, для відображення інтерфейс модель даних продукту наведена у лістингу 3.2.

Лістинг 3.2 — Модель продукту у коді

```
export interface Card {
```

```
  id: string;
  images: [string, string?];
  title: string;
  rating: number;
  price: number;
  brand: string;
  category: string;
  isCart: boolean;
  isWished: boolean;

}
```

3.1.2 Список товарів

Для формування списку продуктів необхідно створити окремий компонент, який буде цим займатись. Тому знову задопомогою терміналу створюємо компонент CardList. Пропишемо декоратор @Input() для отримання масиву з об’єктами продуктів у поле currentPageProducts. У лістингу 3.3 наведено код класу компонента CardListComponent.

Лістинг 3.3 — Клас компонента CardListComponent

```
export class CardlistComponent {
  @Input() currentPageProducts = [];
}
```

Цей компонент буде приймати масив об’єктів та задопомогою директиви *ngFor буде формувати список компонент продуктів. Далі скористаємося вище згаданою директивою для рендеру списку компонент. Важливо огорнути компоненти продуктів тегом <ul>, цей тег сеціально створений для списків так браузер та скрінрідери будуть розпізнавати цей список та окремо виділяти його для користувача. Код html шблону компонент наведено у лістингу 3.4.

Лістинг 3.4 — Код шаблону компонента CardListComponent

```
‹ul class="products__inner">
   <li *ngFor="let card of currentPageProducts">
    <app-card [card]="card"></app-card>
   </li>
</ul>
```

Скріншот із виглядом списку товарів показаний на рисунку 3.3.

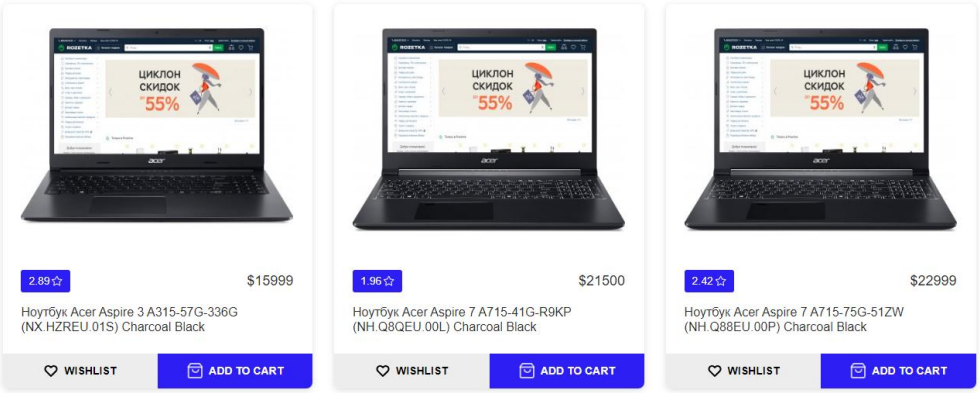

Рисунок 3.3 — Готовий компонент списку товарів

Поки що компоненти використовують статичні дані. Тематика цієї роботи зосерджена на роботі із методами покращення доступності та їх використання на практиці, тому ми будемо робити емуляцію сервера, тобто логіку сервера, але без бази даних.

Перейдемо до логіки отримання товарів на стороні клієнта. Щоб релізувати таку логіку нам потрібно познайомитись із ще одним видом сутностей у Angular, а сааме із сервісами. Сервіс повинен робити щось конкретне і працювати тільки над цим. Опишемо у сервісі метод для запиту на увесь спсок товарів. Код методу нведений у лістингу 3.5.

Лістинг 3.5 — Метод із запитом повного списку продуктів.

```
getAllProducts(): Observable<any> {
  return this.httpClient.get(this.cardURL);
```

```
  }
```

Залишилось тепер використати наш метод у компоненті списку карток продуктів, ініціалізуємо екземпляр класу ProductStorageService у конструкторі компоненту класу. Викликаємо метод getAllProducts, цей метод повертає Observable – певна черга із подій, сутність бібліотеки RxJs, на які нам потрібно підпсатись задопомогою методу subscribe. Лістинг коду з класом CardlistContainer наведено у лістнгу 3.6.

Лістинг 3.6 — Клас CardlistContainer

```
export class CardlistContainerComponent implements OnInit {
  currentPageProducts = [];
  constructor(private productService: ProductStorageService){}
  ngOnInit(): void {
    this.filtersSubscription$ = this.filters.subscribe(state => {
      this.productService. getAllProducts (state).pipe(
        take(1)
      ).subscribe(response => {
        const numberOfPages = Math.ceil(response.numberOfProducts / 9);
        this.arrayForPagination = Array.from(Array(numberOfPages).keys());
        this.currentPageProducts = response.products;
      })
    })
  }
  ngOnDestroy() {
    this.filtersSubscription$.unsubscribe();
  }
}
```

Якщо розглянути випадок із великою кількістю продуктів, то користувачу буде важко гортати увесь список продуктів у пошуках бажаного,

тому варто зробити роцес пошуку максимально швидким та зручним. Для цього потрібно створити елемент пошуку товарів.

3.1.3 Компонент пошуку товарів

Наступним кроком напишемо логіку відравки запиту із параметрами пошуку. Створюємо файли для компонента елемента пошуку. Html шаблон міститиме одне поле введення (<input>), яке буде огорнуте у тег <form> із кнокою для підвердження введення. Скріншот із виглядом поля введення на рисунку 3.4.

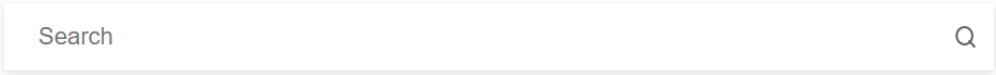

Рисунок 3.4 — Вигляд поля для пошуку

Щоб зчитувати значення нам потрібно повішати на подію підтвердження форми метод, який буде по посиланню зчитувати значення поля вводу та передавати для компонента рівнем вище. Код класу компонента SearchComponent наведено у лістингу 3.7.

Лістинг 3.7 — Клас компонента SearchComponent

```
export class SearchComponent {
  @Output() value: EventEmitter<any> = new EventEmitter();
  getSearhUrl(event:Event, searchText:string):void {
    event.preventDefault();
    this.value.emit(searchText);
  }
}
```

Ініціалізуємо суність BehaviorSubject у змінну filters. У підисці на наш BehaviorSubject будемо виконувати запит для отримання продуктів, таки чином він буде спрацьовувати після кожної зміни BehaviorSubject. А сам BehaviorSubject буде змінюватись у методі onSearch компонента cardlist-container, який спрацьовує при зміні поля пошуку. В результаті ри кожній

зміні масиву товарів, буде спрацьовувати ререндер списку товарів з відфільтрованими товарами. Код оновленого отримання списку товарів у лістингу 3.8.

Лістинг 3.8 — Код оновленого отримання списку товарів

```
this.filtersSubscription$ = this.filters.subscribe(search => {
  this.productService.getAllProducts(search).pipe(
    take(1)
  ).subscribe(productList => {
    this.currentPageProducts = productList;
  })

})
```

Тепер користувач зможе знайти по назві потрібний йому товар, але що якщо він захоче проглянути потрібні йому товари певного бренду або категорії? Для цього необхідно зробити фільтри. Щоб при виборі бренду або категорії товри фільтрувалися.

Спочатку потрібно реалізувати форму фільтру, а потім логіку фільтрації. Бренди та категорії будуть приходити із сервера і тоді клієнт має відобразити їх у формі фільтрації.

Почнемо розробку форми фільтрації. Створюємо компонент з відповідними файлами під назвою sidebar. Цей компонент буде обгорткою для нашої форми. Він буде отримувати значення усіх брендів та категорій з сервера, перетворювати у об’єкти для відображення елеметів <input type=”checkbox”>. Код класу компонента SidebarComponent наведено у лістингу 3.9.

Лістинг 3.9 — Код оновленого отримання списку товарів

```
export class SidebarComponent implements OnInit {
  @Output() checkedFilters: EventEmitter<any> = new EventEmitter();
  filters: any = {brands: [], categories: []}
```

```
  constructor(private filtersService: FiltersService) { }
  ngOnInit(): void {
    this.filtersService.getCategories().subscribe(categories => {
      this.filters.categories = categories.map((item: string) => ({ value: item, checked:
false}));
    });
    this.filtersService.getBrands().subscribe(brands => {
      this.filters.brands = brands.map((item: string) => ({ value: item, checked:
false}));
    });
  }
  onCheck(event: Array<string>):void {
    this.checkedFilters.emit(event);
  }
}
```

3.1.4 Форма фільтрації товарів

Компонент форми фільтрації буде отримувати дані для рендеру чекбоків та обробляти вибір користувачем того чи іншого параметру. Для відслідковування вибору користувача, опишемо метод onCheck, який буде виконуатись при тригері чекбоксів і буде передавати інформацію про обраний елемент батьківському компоненту sidebar. Реалізовувати процес передачі інформації на верхній рівень будемо за допомогою класу EventEmitter та декоратору @Output(). Клас FilterFormCompoent наведено у лістингу 3.10.

Лістинг 3.10 — Код класу FilterFormCompoent

```
export class FilterFormComponent {
  @Output() filtersChecked: EventEmitter<any> = new EventEmitter();
  filtersKeys: any = [];
  @Input() filters: any = {brands: [], categories: []}
  ngOnInit(): void {
```

```
    this.filtersKeys = Object.keys(this.filters);
  }
  onCheck(input: any):void {
    const type = this.getType(input.value);
    this.uncheckOtherInputs(type, input.value);
    this.filtersChecked.emit({
      value: !input.checked ? input.value.toLowerCase().replace(/ /g, '_') : '',
      type
    });
  }
  getType(value: string): string {
    const filtersTypes = Object.keys(this.filters);
    return filtersTypes.filter(key =>
      this.filters[key].find((item: any) => item.value === value)
    )[0];
  }
  uncheckOtherInputs(type: string, value: string){
    this.filters[type] =  this.filters[type].map((item: any)  =>
      item.value !== value ?
       {...item, checked: false} :
       {...item, checked: !item.checked}
    );
  }
}
```

Метод getType визначає тип фільтру, тобто чи це фільтр брендів або категорій, а метод uncheckOtherInputs вимикає всі інші <input> одного типу.

Опишемо логіку відправки запиту із потрібними параметрами у тілі запиту. Почнемо із того, що зараз компонет sidebar ніяк не опрацьовує і нікуди не передає дані про активовані фільтри. Нам же потрібно, як і по аналогії реалізації пошуку передавати дані про зміни у cardlist компонент та описати

там логіку реагування на зміни у фільтрах через екземпляр класу BehaviorSubject. Тому знову використовуєм EventEmitter та @Output() у компоненті sidebar та передаєм у cardlist. Код поля із типом BehaviorSubject наведений у лістнгу 3.11.

Лістинг 3.11 — Код класу FilterFormCompoent

```
filtersState: any = {
  q: '',
  categories: '',
  brands: '',
  page: 1,
  limit: 9,
}
filters = new BehaviorSubject(this.filtersState);
```

У cardlist компоненті описуєм метод, який буде реагувати на зміни фільтрів onCheck. Тепер коли, логіка запитів реагує на зміни фільтрів оновимо метод сервісу ProductStorageService, та напишемо логіку формування параметрів тіла запиту. Параметри функція буде отримувати у вигляді об’єкта і за допомогою метода Object.entries() та методу масивів map будемо додавати кожен парметр у об’єкт параметрів, методом для об’єктів Http параметрів append(). Код оновленого методу для запиту товарів описаний у лістингу 3.12.

Лістинг 3.12 — Код оновленого методу для запиту товарів

```
 getFilteredProducts(paramsInfo: string): Observable<any> {
    let params = new HttpParams();
    Object.entries(paramsInfo).map(([key, value]) => {
      params = params.append(key, value)
    })
    return this.httpClient.get('http://localhost:5000/search', { params: params });
  }
```

Готовий функціонал дозволяє повністю створити домашню сторінку додатку. Тому створюємо файли компоненту домашньої сторінки, додаємо у шаблон сторінки компонент cardlist-container, також наперед створимо компоненти сторінок корзини товарів, списку бажаних товарів користувача, окремої сторінки кожного товару та сторінки у випадку використання некорректного шляху в додатку.

Нам необхідно створити окремий модуль для навігації у якому ми опишемо усі можливі маршрути у додатку. У модулі навігації описуємо маршрути – це массив об’єктів, що відповідає типу Routes. Кожен маршрут має містити поле path – шлях маршруту та component - компонента, яка буде відтворюватись при вказаному path у адресному рядку. Повні розписані маршрути наведені у лістингу 3.13.

Лістинг 3.13 — Маршрути навігації додатку

```
const routes: Routes = [
  {
    path: '',
    component: HomeComponent,
  },
  {
    path: 'wishlist',
    component: WishlistComponent,
  },
  {
    path: 'cart',
    component: CartComponent,
  },
  {
    path: 'products/:id',
    component: SingleProductComponent,
```

```
  },
  {
    path: '**',
    component: NotFoundComponent,
  },

];
```

3.1.5 Навігація клієнта

Останнім кроком для розробки модуля навігації буде використання методу forRoot классу RouterModule. Метод RouterModule.forRoot(Routes) створює та налаштовує модуль з усіма провайдерами та директивами маршрутизатора. Додатково налаштовує слухач програми для виконання початкової навігації. Залишилось лише імпорутвати наш AppRoutingModule у app модуль та у шаблоні app компонента прописати компонент <router-outlet>, який надає RoutingModule для використання нашої навігації у додатку. Код класу RouterModule наведено у лістингу 3.14.

Лістинг 3.14 — Код класу RoutingModule

```
@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule],
})
export class AppRoutingModule {}
```

Почнемо із окремої сторінки кожного товару. Для переходу на такі сторінки кожна картка товару має мати посилання, яке переведе користувача на сторінку цього товару. Тому зробимо назву товару посиланням за допомогою директиви [routerLink] від RouterModule. Як значення присвоїмо масив, першим елементом якого буде стрічка «products», а другим id товару. Таким чином, у кожної сторінки продукту буде свій унікальний url.

Все що нам необхідно це здійснити запит для отримання одного товару та відмалювати інформацію із полів об’єкту товару у html шаблоні. Код запиту наведено у лістнгу 3.15.

Лістинг 3.15 — Код класу RoutingModule

```
getFilteredProducts(id: number): Observable<any> {
  return this.httpClient.get(`http://localhost:5000/products/${id}`);
}
```

При переході на сторінку, url кожної сторінки міститиме унікальний id товару, тому ми цим і скористаємось. Дістанемо id із url з допомогою класу ActivatedRoute, який надає нам Angular. Цей клас може повертати певну інформацію про поточний роут користувача. Підисавшись на поле params класу ActivatedRoute, можна дістати id, яке ми передаємо в url при переході на сторінку. Записавши значення id у поле компонента з такою назвою, можна здійснити запит із необхідним параметром. Код отримання Id продукту з url наведено у лістингу 3.16.

Лістинг 3.16 — Код класу RoutingModule

```
  ngOnInit(): void {
    this.activatedRoute.params
      .pipe(takeUntil(this.destroy$))
      .subscribe((params) => (this.id = params.id));
    this.getSingleProduct();
  }
```

Опишемо метод аналогічний до методу отримання усіх продуктів getSingleProduct, метод для отримання одного продукту буде відрізнятись тільки параметрами запиту та url для запиту від інших запитів на сервер.

У методі життєвого циклу компонентів Angular ngOnInit викличемо новостворений метод getSingleProduct та підпишемось на Observable, який повертає метод. Останнім кроком, присвоївши дані про товар у відповідне

поле у компоненті сторінки, скориставшись інтероляцією виводимо дані про товар у відовідні html елементи. Код методу getSingleProduct на ведений у лістингу 3.17.

Лістинг 3.17 — Код методу getSingleProduct

```
getSingleProduct() {
  this.productService
    .getSingleProduct(this.id)
    .pipe(takeUntil(this.destroy$))
    .subscribe((card) => {
      return (this.card = card[0]);
    });
}
```

Сторінки корзини та бажаних товарів будуть відрізнятись тільки в url для запитів, тому розглянемо логіку корзини. Логіка клієнта має робити запит для отримання списку бажаних товарів на сервер. Тому створюємо сервіс із назвою WishProductsStorage.

Для отримання списку бажаних товарів у компонент, необхідно у методі ngOnInit компонента викликати метод getWishlist та у підписці на Observable, який повертає метод, присвоїти полю wishlist компонента дані, які надійшли з сервера. Далі передаєм значення поля wishlist у cardlist компонент, який відрендерить увесь список товарів. Код класу WishlistComponent наведено у лістингу 3.18.

Лістинг 3.18 — Код класу RoutingModule

```
export class WishlistComponent implements OnInit, DoCheck {
  wishlist: Card[] = [];
  constructor(private wishlistService: WishlistService, ) {}
  ngOnInit(): void {
    this.wishlist = this.wishlistService.getWishlist().pipe(
```

```
      take(1)
    ).subscribe(response => {
      this.wishlist = response;
    });
  }
}
```

Запит для оновлення статусу у продуктах буде спрацьовувати при клікові на кнопку у картці товару і поралельно змінювати своє локальне поле isWished для відображення змін у картці товару для користувача.

3.2 Розробка серверної частини додатку

Для зручності розробки варто встановити інструменти nodemon та body-parser. Тож встановимо ці пакети через npm. Для того щоб інструмент працював у файлі package.json, у властивості “scripts” потрібно вказати команду запуску основного файлу. Код із конфігурацією команди наведений у лістингу 3.19.

Лістинг 3.19 — Конфігурція команди запуску основного файлу сервера

```
"scripts": {
      "start": "nodemon server/app.js"
   },
```

Наступним кроком необхідно встановити сам фреймворк для розробки ExpressJS задопомогою npm.

Першим кроком буде імпортування самого фреймворку у змінну та створюємо об’єкт застосунку та пропишемо слухача нашого сервера. Далі скористуємося body-parser, що дозволить зручніше працювати із тілами запитів. Початковий код головного файлу сервера наведено у лістингу 3.20.

Лістинг 3.20 — Початковий код головного файлу сервера

```
const express = require("express");
const bodyParser = require("body-parser");
```

```
const cors = require("cors");
const app = express();
const port = 5000;
app.use(cors());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());
app.listen(port, () => {
   console.log(port)
});
```

3.2.1 Логіка обробки товарів

Для початку створимо директорію для контроллерів продуктів, методів, які будуть спрацьовувати у разі запиту на роут із url /products. Нам необхідні контроллери для відправки відфільтрованих продуктів та для відправки одного окремого продукту на клієнт. Для відправки відфільтрованих продуктів створюємо функцію getFilteredProducts. Ця функція буде зчитувати query параметри запиту: id, категорія, бренд та q та на основі цих переметрів фільтрувати увесь доступний список об'єктів за відповідними полями. Код контроллера getFilteredProducts наведено у лістингу 3.21.

Лістинг 3.21 — Код контроллера getFilteredProducts

```
const getFilteredProduts = (req, res) => {
  const query = req.query.q;
  const brands = req.query.brands;
  const categories = req.query.categories;
  const minPrice = req.query.minPrice;
  const maxPrice = req.query.maxPrice;
  const minRating = req.query.minRating;
  const maxRating = req.query.maxRating;
  let products;
  fs.readFile(productsDb, "utf8", (err, data) => {
    if (err) throw err;
```

```
else {
  products = JSON.parse(data);
  let filteredProducts = products.filter((product) => {
    const isQuery = query ? product.title.includes(query) : true;
    const isBrand = brands ? product.brand === brands : true;
    const isCategory = categories ? product.category === categories : true;
    const isInPriceRange =
      minPrice && maxPrice
        ? product.price <= +maxPrice && product.price >= +minPrice
        : true;
    const isInRatingRange =
      minRating && maxRating
        ? product.rating <= +maxRating && product.rating >= +minRating
        : true;
    if (
      isQuery &&
      isBrand &&
      isCategory &&
      isInPriceRange &&
      isInRatingRange
    ) {
      return product;
    }
  });
  filteredProducts = paginationServices.paginationResult(
    filteredProducts,
    page,
    limit
  );
  filteredProducts = JSON.stringify(filteredProducts);
```

```
    res.send(filteredProducts);
   }
  });
};
```

Функція getSingleProducts матиме простішу логіку, вона отримуватиме із параметрів запиту тільки id та на його основі знаходити потрібний продукт порівнюючи з id товару. Код getProduct наведено у лістингу 3.22.

Лістинг 3.22 — Код контроллера getProduct

```
const getProduct = (req, res) => {
  const { id } = req.params;
  fs.readFile(productsDb, "utf8", (err, data) => {
    if (err) throw err;
    else {
      let products = JSON.parse(data);
      let product = products.find((product) => product.id === id);
      product = JSON.stringify(product);
      res.send(product);
    }
  });
};
```

Після реалізації контроллерів необхідно викликати їх при тригері відповідних роутів, імпортуємо їх у файл для роутів товарів та передаєм їх у якості callback функцій у роути. Код файлу з роутами товарів наведено у лістингу 3.23.

Лістинг 3.23 — Код файлу із роутами товарів

```
const express = require("express");
const router = express.Router();
const productsController = require("../controlers/products");
```

```
router.get("/", productsController.getAllProducts);
router.get("/number", productsController.getNumberOfAllProducts);
router.get("/:id", productsController.getProduct);
module.exports = router;
```

3.2.2 Логіка обробки категорій та брендів

Наступним кроком йде реалізація логіки отримання брендів та категорій. Створюємо файл для опису контроллера відправки усіх брендів. Усе що буде робити цей контроллер діставати список брендів із сховища та передавати їх на клієнт. Імпортуємо контроллер у файл з роутом брендів та передаєм у якості callback. Реалізація контроллера категорій та використання аналогічне як і для брендів. Код контроллера для відправки брендів наведено у лістингу 3.24.

Лістинг 3.24 — Код контроллера для відправки брендів

```
const getAllBrands = (req, res) => {
  fs.readFile(brandsDb, "utf8", (err, data) => {
    if (err) throw err;
    else {
      let brands = JSON.parse(data);
      brands = JSON.stringify(brands);
      res.send(brands);
    }
  });
};
```

3.2.3 Логіка обробки списку бажаних товарів

У додатку є функція додавання товару у список бажаних товарів. Для початку нам необхідно додати логіку додавання продукту у список. Напишемо контролер postWishProduct функція зчитуватиме id продукту знаходити із списку усіх товарів необхідний та змінювати його поле isWished на true та відправляти на клієнт. Код контроллера наведений у лістингу 3.25.

Лістинг 3.25 — Код контроллера postWishProduct

```
const postWishProduct = (req, res) => {
  const { id } = req.params;
  let products;
  fs.readFile(productsDb, "utf8", (err, data) => {
    if (err) throw err;
    else {
      products = JSON.parse(data);
      products = products.map((productFromDb) =>
        productFromDb.id === id
          ? { ...productFromDb, isWished: true }
          : productFromDb
      );
      products = JSON.stringify(products);
    }
    fs.writeFile(productsDb, products, (err) => {
      if (err) throw err;
      else {
        res.send(products);
      }
    });
  });
};
```

Якщо ми реалізували логіку додавання продукту і список бажаних товарів, то є необхідність у функції видалення товару. Створюємо контроллер deleteWishProduct, він буде працювати за принципом postWishProduct функції, але змінювати поле isWished на значення false.

Тепер коли ми маєм можливість зміни списку бажаних товарів, останнім кроком варто описати контроллер для відправки усього списку бажаних товарів на клієнт. Функція getWishProducts буде зчитувати із сховища увесь

список товарів, фільтрувати за полем isWished та повертати на клієнт. Код контроллера getWishProducts наведено у лістингу 3.26.

Лістинг 3.26 — Код контроллера getWishProducts

```
const getWishProducts = (req, res) => {
  const { page, limit } = req.query;
  fs.readFile(productsDb, "utf8", (err, data) => {
    if (err) throw err;
    else {
      let products = JSON.parse(data);
      products = products.filter(({ isWished }) => isWished === true);
      products = paginationServices.paginationResult(products, page, limit);
      products = JSON.stringify(products);
      res.send(products);
    }
  });
};
```

Імпортуємо створені контролери у файл для роутів списку бажаних продуктів та передаємо у відповідні роути у якості callback функцій. Отримані роути необхідно імпортувати у головний файл та використати. Код використання роутів наведений у лістингу 3.27.

Лістинг 3.27 — Код використання роутів

```
const products = require("./routes/products");
const brands = require("./routes/brands");
const categories = require("./routes/categories");
const search = require("./routes/search");
const wishProducts = require("./routes/wish-products");
app.use("/products", products);
app.use("/brands", brands);
```

```
app.use("/categories", categories);
app.use("/search", search);
app.use("/wish-products", wishProducts);
```

4 ТЕСТУВАННЯ ДОДАТКУ

4.1 Тестування доступності додатку

Доступність важко протестувати повністю або ж в рельних умовах, але є кілька кроків, зробивши які можна знайти помилки та завчасно їх виравити.

Перевіримо альтернативний текст на наявність у зображенях та іншону нетекстовому вмісту. Усі зображення та інший нетекстовий вміст повинні мати текстову альтернативу. Щоб такі допоміжні технології могли належним чином інтерпретувати та передавати такі об’єкти, як графіка, їм потрібен точний альтернативний текст. Це тому, що ці інструменти не можуть природним чином прочитати те, що відображається в нетекстовому вмісті, або його призначення [17].

Щоб перевірити текст заміщення, ви можете використовувати програму зчитування з екрана або іншу допоміжну технологію на комп’ютері та мобільному пристрої. Якщо ви технічно підковані й схильні до цього, ви можете переглянути код і перевірити, чи є описи в атрибутах alt зображення, наприклад. Ви також можете надіслати запит на безкоштовне та конфіденційне сканування вашого веб-сайту, яке виявить відсутній альтернативний текст. Проте зауважте, що хоча автоматизований інструмент може визначити, чи є на зображенні альтернативний текст, він не може визначити, чи він доречний чи достатній.

У нашому додатку картинки використовуються тільки у картках товарів, картки товару генеруються залежно від кількості товарів та у alt атрибут картинки записується категорія до якої належить товар, тобто ноутбук, монітор і т.д. Тому на усіх картинках буде атрибут alt і він завжди буде нести правильну інформацію.

Перевірка субтитрів та стенограм на відео. Субтитри та стенограми життєво важливі для доступності медіа та мультимедіа, як-от відео. Титри — це текстові альтернативи аудіоконтенту, синхронізовані з відео. Вони мають містити голосові діалоги, релевантні звуки та інші контекстуальні елементи,

як-от музика, які можуть мати вирішальне значення для досягнення повної мети чи відчуття від відео. Багатьом людям перш за все спадають на думку переваги субтитрів для глухих або людей із втратою слуху, і, безперечно, з цієї причини вони потрібні. Однак вони також є корисними для всіх у різний час, наприклад у тихій обстановці або коли читання та перегляд разом допомагають зрозуміти [17].

Стенограми — це текстові версії відеовмісту, тому вони мають містити всі вимовлені слова та важливі звуки, а також текстовий опис усього важливого, що візуально відображається у відео. Стенограми також можуть бути великим прискорювачем SEO.

Перевірити субтитри та стенограми легко. Перейдіть до відео та в програвачі подивіться, чи є кнопка чи опція для ввімкнення субтитрів (якщо субтитри відкриті, вони з’являться автоматично, і їх не можна вимкнути). Переконайтеся, що кнопка працює з мишею та клавіатурою. Потім визначте, чи є текстова розшифровка, яка супроводжує відео. Якщо ви не знайшли їх або вони здаються вам неадекватними, настав час спробувати зробити відео надійнішими та доступнішими. У додатку відсутній контент такого роді тому у нашому випадку цей пункт немає необхіднсті.

Перевіримо контрастність кольорів. Для цифрової доступності контраст кольорів настільки ж важливий, як і простий. Кольоровий контраст означає різницю в освітленості між шрифтом (або чим-небудь на передньому плані) і його фоном. Завдяки використанню досить контрастних кольорів видимість шрифту веб-сайту є достатньо чіткою, щоб її розрізнила більшість людей. Відвідувачі веб-сайту зі слабким зором, низькоконтрастним зором або дефіцитом сприйняття кольорів отримують особливу користь, коли вміст має достатній контраст, і більшість людей (навіть із сильним зором) цінують, що їм не доводиться надмірно напружувати очі, щоб прочитати матеріал.

Коли ми говоримо про коефіцієнти контрастності, ми говоримо про фактичне числове значення, яке визначає рівень контрасту. Критерій успіху 1.4.3 рекомендацій щодо доступності веб-вмісту (WCAG) стверджує, що

звичайний текст має відповідати мінімальному коефіцієнту контрастності принаймні 4,5 до 1, а великий текст (18 пунктів або більше, або 14 пунктів або більше та жирний) має відповідати мінімальному контрасту співвідношення не менше 3 до 1.

Виконання вимог щодо контрастності кольорів справді є одним із найважливіших аспектів доступності, і часто це питання, яке можна легко виправити. У додатку використовуються три комбінації кольорів. Перша комбінція фоновий #fff та текстовий #000, рівень контрасту яких становить 1 до 21. Наступна комбінація фоновий #090086 та #fff, рівень контрасту становить 1 до 15,48. Останні будуть фоновий #ededed та текстовий #000 їх рівень контрасту дорівнює 1 до 17,93. Цей крок також є успішним тому що кольорова палітра додатку відповідає усім необхідним стандартам. Повна палітра використаних кольорів зображена на рисунку 4.1.

Рисунок 4.1 — Палітра використаних кольорів

Тестування клавіатури. Використовуючи типові клавішні команди, як-от клавіші Tab і Shift-Tab, ви можете почати отримувати уявлення про стан доступності вашого веб-сайту. Якщо ви помітили, що є певні елементи, до яких ви не можете дістатися, або їх легко загубити на сторінці, можливо, виникли проблеми з доступністю клавіатури.

Багато людей не можуть або не хочуть використовувати мишу для навігації в Інтернеті, а замість цього використовують клавіатуру, емулятор клавіатури чи інший альтернативний пристрій введення. З цієї причини важливо, щоб кожне посилання, елемент керування та функція, якими можна керувати за допомогою миші, були доступні лише за допомогою клавіатури. Крім того, має бути чітка візуальна індикація поточного елемента у фокусі,

щоб відвідувачі веб-сайту знали, де вони знаходяться на сторінці та яке посилання чи елемент керування вони можуть вибрати. Це поширюється на всі функції, включаючи можливість вибору в спадних меню, а також заповнення та надсилання форм.

Протестуємо керування сайтом задопомогою програми зчитування екрану та клавітури без використання миші. Кожен елемент керування, такий як праорці фільтрів, пошук, перехід на інші сторінки, елементи пагінації є доступними для користування сайтом з допомогою клавіатури. Кожем з цих елементів можна обрати та зробити дію над ним, тобто обрати прапорець або активувати пошук певних товарів о назві. Інформація про товар: ціна, рейтинг, назва також зчитуються та озвучуються програмою зчитування екрану. Змістовні aria-label та правильні tabindex атрибути дозволили зробити процес освоєння сайту з програмами зчитування екрану досить зручним.

Переконайтеся, що ваш сайт можна масштабувати без втрати вмісту чи функціональності. Відповідно до вимог WCAG вміст можна збільшити до 200% і працювати без допоміжних технологій. Крім того, збільшення екрана не повинно заважати іншим вимогам доступності.

На щастя, перевірити це до певної міри може бути досить легко. Збільште веб-браузер до 200% і подивіться, що станеться з вмістом і макетом веб-сторінки. Чи помічаєте ви, що елементи вмісту накладаються або зникають, чи вони добре складаються та регулюються? Чи можна виконувати всі завдання за допомогою миші та клавіатури? Чи добре працюють елементи навігації та меню? Виконання цього попереднього тестування не є вичерпним чи комплексним, але воно може допомогти вам визначити деякі очевидні проблеми з доступністю дисплея.

Збільшивши масштаб сторінки до 200% не було помітно ніяких змін, окрім змін у стилях для комфортного відображення контенту. Усі фільтри залишись на сторінці та коректно функціонують, нявний пошук, який також функіонує за призначення, картки товарів не втратили контент та добре

відображаються скріншот із збільшеним масштабом сторінки додатку до 200% показано на рисунку 4.2.

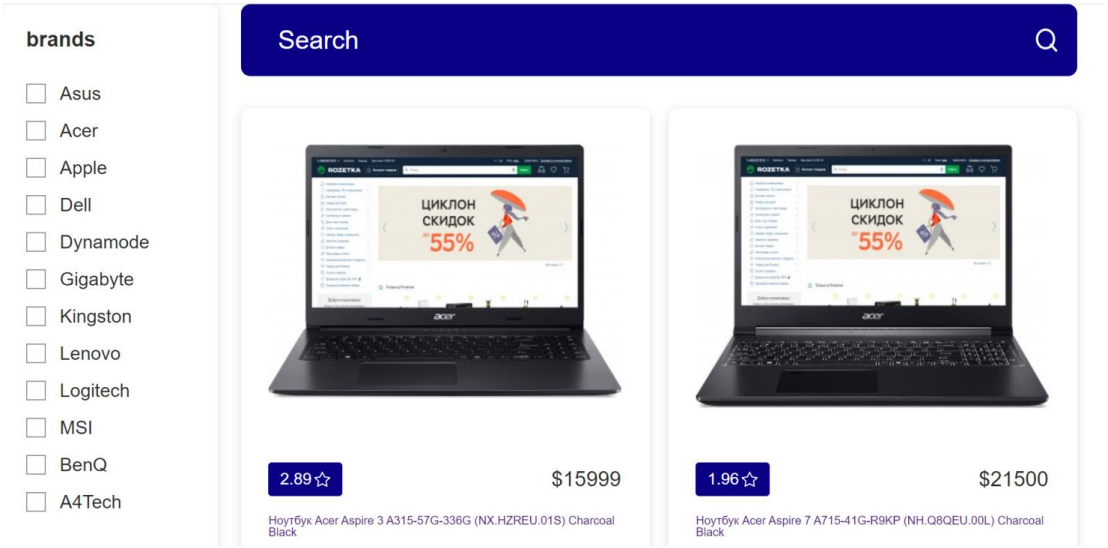

Рисунок 4.2 — Скріншот із збільшеним масштабом сторінки додатку до 200%

Останнім етапом тестування має бути перевірка додатку сервісом lighthouse, щоб можна було переконатись у тому, що наш сайт у параметрі доступності перевершує розглянуті у роботі аналоги. Після огляду додатку інструментом результат за значення доступності дорівнює 100%, що є зразковим результатом і підтверджує усіх тестування вищеописаних кроків.

4.2 Інструкція користувача

Враховуючи, що розроблений продукт являє собою веб-додаток, вимоги для його запуску та використання мінімальні. Додаток можна відкрити з більшості носіїв, більшості операційних системи, де є браузер. Якщо точніше, то мінімальні та рекомендовані вимоги до системи популярних браузерів наведені у таблиці 4.1.

Таблиця 4.1 — рекомендовні системні вимоги браузерів Chrome, Opera, Safari

	Chrome	Opera	Safari
Процесор Windows	Pentium 4	Pentium II	500-MHz
	Chrome	Opera	Safari
Процесор Mac OS	Pentium 4	Pentium II	500-MHz Pentium-class
Рекомендована оперативна пам’ять	512Мб	512Мб	512Мб
Мінімальне місце на носії	100Мб	20Мб	100Мб
Рекомендоване місце на носії	100Мб	100Мб	100Мб
Мінімальна версія Windows	Windows XP SP2	Windows XP SP2	Windows XP SP2
Мінімальна версія OS X	OS X 10.5.6	Mac OS X 10.5	OS X 10.5.8
Мінімальна версія Linux	Ubuntu 10.04 Debian 6 OpenSuse 11.3 Fedora Linux 14	Будь-який останній	Недоступний

Якщо говорити про карту навігації додатку то вона невелика, містить всього п’ять елементів:

— домашня сторінка;

— сторінка корзини;

— сторінка бажаних товарів;

— сторінка товару;

— сторінка помилки у випадку незнайденої сторінки за вказаним url у адресному рядку.

Схема навігаційної карти додатку зображена на рисунку 4.3.

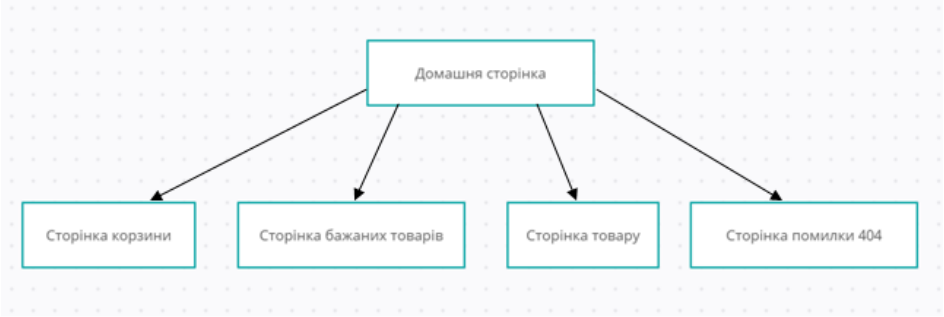

Рисунок 4.3 — Схема навігаційної карти додатку

Перехід між сторінка можна здійснити з допомогою посилань на сторінках. На домашній сторінці можна здійснити перехід на всі достуні в додатку сторінки.

Для сторінок корзини та бажаних товарів виділено спеціальні кнопки на сторінці, зображені вони на рисунку 4.4.

Рисунок 4.4 — Кнопки навігації для сторінок корзини та бажаних товарів

Перша кнопка для переходу на корзини, інша для переходу на сторінку бажаних товарів. Якщо перейти на сторінку бажаних товарів, то з неї можна навігуватися тільки назад на домшню сторінку та у корзину, кнопки навігації сторінки бажаних товарів зображені на рисунку 4.5.

При переході у корзну схожі ситуація, знеї можна навігуватись тільки на домашню сторінку та на сторінку бажаних товарі. Кнопки навігації сторінки корзини зображені на рисунку 4.6.

Рисунок 4.5 — Кнопки навігації сторінки бажаних товарів

Перехід на окрему сторінку певного товару можна здійснити, клікнувши по назві потрібно товару у загальному списку, тому що сама назва і є посиланням на сторінку.

Рисунок 4.6 — Кнопки навігації сторінки корзини

Приклад посилання на картці товару виділено на рисунку 4.7.

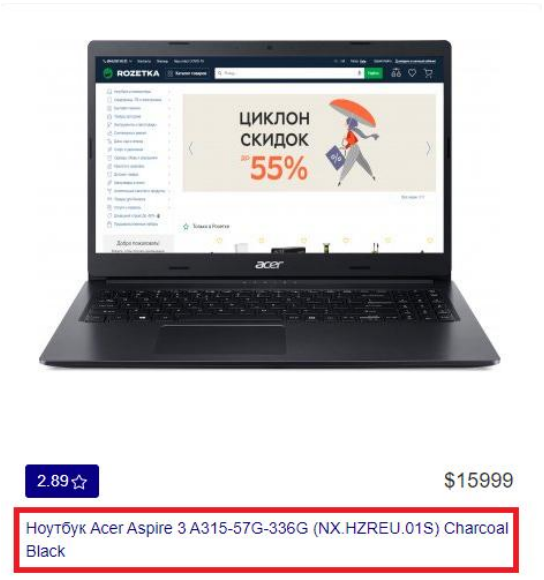

Рисунок 4.7 — Приклад посилання для навігація в окрему сторінку товару

Додавання товарів у корзину та до списку бажаних товарів здійснюється з домогою кнопок, які містить кожна картка товару як і в звгвльному списку так і в окремій сторінці. Вигляд кнопок зображено на рисунку 4.8.

Рисунок 4.8 — Кнопки додавання товарів у корзину та до списку бажаних товарів

Також у додатку реалізований пошук товару по назві товару. Сам елемент зображений на рисунку 4.9.

Рисунок 4.9 — Поле введення для пошуку товару

Варте зауважити те, що пошук чутливий до регістру тому часто назви варто вписувати починаючи з букви верхнього регістру. Початок пошуку товарів можна підтвердити, натиснувши кнопку вводу або натиснувши мишкою по картинці лупи з правого краю елемента.

Для зручності використання було розроблено елемент пагінації товарів. Тобто одна сторінка товарів містить лише дев'ять товарів. Зроблена дана функція булла для уникнення великого скролу на сторінці. Компонент пагінації зображений на рисунку 4.10.

Рисунок 4.10 — Компонент пагінації

Керувати пагінацією можна з допомогою бокових кнопок, так і з допомогою списку номерів сторінок. Кількість сторінок залежить від кількості доступних товарів.

Загальне число знайдених товарів виводиться зверху над полем пошуку товарів. Це значння змінюється залежно від результатів пошуку товарів та від

рзультів фільтрації товарів. Елемент з числом кількості товарів відображений на рисунку 4.11.

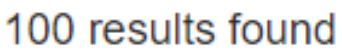

Рисунок 4.11 — Елемент з числом кількості товарів

Якщо ж не буде підібрано жодного товару або не буде доступних товарів взагалі, повідомлення зміниться на «no result found».

Для фільтрування товарів за певними параметрами, а саме брендами та категоріями, додано форму фільтрації. Список фільтрів за брендами показаний на рисунку 4.12.

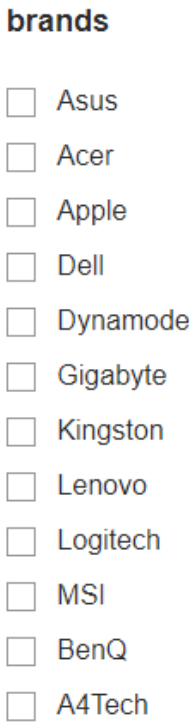

Рисунок 4.12 — Список фільтрів за брендами

Список фільтрів з брендами товарів знаходиться одразу над списком категорій, а сам список зображений на рисунку 4.13.

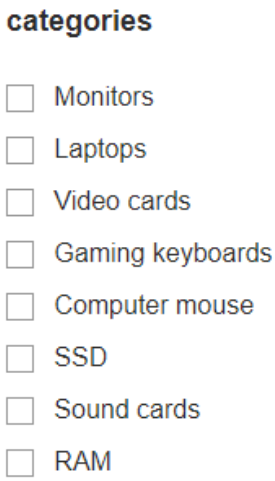

Рисунок 4.13 — Список фільтрів закатегоріями товарів

Обрати можна тільки по одному фільтру у кожному списку, зроблено це для чіткості групування. Групувати можна окремо як і по брнду так і по категорії.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Комерційний та технологічний аудит науково-технічної розробки

Метою даного розділу є проведення технологічного аудиту, в даному випадку нових комп'ютеризованих методів адаптації Web-додатків для використання людьми з обмеженими можливостями. Кінцевий продукт — це Інтернет-магазин, адаптований під потреби людей із обмеженими можливостями.

Особливістю програми є те, що дана технологія надасть можливість користування продуктом людьми із інвалідністю, тобто людьми із проблемами зору, дальтонізном і т.д.

Аналогом може бути такий веб-ресурс як https://www.vodospad.net.ua/, приблизною вартість у 40000 грн.

Для проведення комерційного та технологічного аудиту залучають не менше 3-х незалежних експертів. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням п'ятибальної системи оцінювання за 12-ма критеріями.

5.2 Прогнозування витрат на виконання науково-дослідної (дослідно-конструкторської) роботи

Основна заробітна плата розробників, яка розраховується за формулою:

$$A = \frac{M}{Tp} * t \qquad (5.1)$$

де М — місячний посадовий оклад конкретного розробника (дослідника), грн.;

T_p — число робочих днів в місяці, 21 днів;

t — число днів роботи розробника ,

Результати розрахунків зведемо до таблиці 5.1.

Таблиця 5.1 — Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	38000	1809,52	38	68761,905
Програміст	34000	1619,05	38	61523,810
Всього				130285,71

Так як в даному випадку розробляється програмний продукт, то розробник виступає одночасно і основним робітником, і тестувальником розроблюваного програмного продукту.

Додаткова заробітна плата розробників, які приймали участь в розробці обладнання.

Додаткова заробітна плата прийнято розраховувати як 11,5 % від основної заробітної плати розробників та робітників:

$$З_д = З_о \cdot 11,5\ \% / 100\ \% \qquad (5.2)$$

$$З_д = (130285,71 \cdot 11,5\ \% / 100\ \%) = 14982,86\ \text{(грн.)}$$

Нарахування на заробітну плату розробників.

Згідно діючого законодавства нарахування на заробітну плату складають 22 % від суми основної та додаткової заробітної плати.

$$Н_з = (З_о + З_д) \cdot 22\ \% / 100\% \qquad (5.3)$$

$$Н_з = (130285,71 + 14982,86) \cdot 22\ \% / 100\ \% = 31959,09\ \text{(грн.)}$$

Витрати на матеріали і комплектуючі. Оскільки для розроблювального пристрою не потрібно витрачати матеріали та комплектуючи, то витрати на матеріали і комплектуючі дорівнюють нулю.

Амортизація обладнання, яке використовувалось для проведення розробки.

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді амортизація обладнання, що використовувалась для розробки розраховується за формулою:

$$А = \frac{Ц}{Тв} * \frac{t}{12} \quad (5.4)$$

де Ц — балансова вартість обладнання, грн.;

Т — термін корисного використання обладнання згідно податкового законодавства, років;

$t_{вик}$ — термін використання під час розробки, місяців,

Розрахуємо, для прикладу, амортизаційні витрати на комп'ютер балансова вартість якого становить 30000 грн., термін його корисного використання згідно податкового законодавства – 2 роки, а термін його фактичного використання – 1,81 міс.

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до таблиці 5.2.

Вартість ліцензійної ОС та спеціалізованих ліцензійних нематеріальних ресурсів безкоштовна (NonVisual Desktop Access (NVDA), Visual studio code).

Витрати на силову електроенергію. Тарифи на розподіл електроенергії для всіх енергорозподільних компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою:

$$Ⅎ_{ⅉ} = Ⅎ \cdot ↿ \cdot ∧ \cdot Ⴝ_{ʧ}, \quad (5.5)$$

де В — вартість 1 кВт-години електроенергії для 1 класу підприємства, В = 6,2 грн./кВт;

П — встановлена потужність обладнання, кВт. П = 0,35 кВт;

Ф — фактична кількість годин роботи обладнання, годин;

$К_п$ — коефіцієнт використання потужності, $К_п$ = 0,9,

$$В_е = 0,9 \cdot 0,35 \cdot 8 \cdot 38 \cdot 6,2 = 593,712 \text{ (грн.)}$$

Таблиця 5.2 — Амортизаційні відрахування матеріальних і нематеріальних ресурсів для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортиза-ційні відрахуван-ня, грн.
Комп'ютер та комп'ютерна периферія (Ноутбук Lenovo IdeaPad Gaming 3 15IHU6 (82K101FJRA) Shadow Black)	30000	2	1,81	2261,905
Офісне обладнення (меблі)	25000	4	1,81	942,460
Приміщення	1200000	20	1,81	9047,619
Всього				12251,98

Інші витрати та загальновиробничі витрати. До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників:

$$, І = (Зо + Зр) * \frac{Н}{100\%} \quad (5.6)$$

де $Н_{ів}$ — норма нарахування за статтею «Інші витрати»,

$$І_в = 130285,71 * 83\% / 100\% = 108137,1 \text{ (грн.)}$$

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$Н = (Зо + Зр) * \frac{Н}{100\%} \quad (5.7)$$

де $Н_{нзв}$ — норма нарахування за статтею «Накладні (загальновиробничі) витрати»,

$$Н_{нзв} = 130285,71 * 144\ \% / 100\ \% = 187611 \text{ (грн.)}$$

Витрати на проведення науково-дослідної роботи.

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

$$В_{заг} = 130285,71+14982,86+31959,09+12251,98+593,71+108137,1+$$
$$+187611 = 485821,92 \text{ грн.}$$

Розрахунок загальних витрат на науково-дослідну (науково-технічну) роботу та оформлення її результатів.

Загальні витрати на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються і „ визначається за формулою:

$$ЗВ = \frac{В}{\eta} \quad (5.8)$$

де η — коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи,

Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η=0,1; технічного проектування, то η=0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7; впровадження, то η=0,9. Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка:

ЗВ = 485821,92 / 0,5 = 971644 грн.

5.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

Припустимо, що при прогнозованій ціні 15000 грн. за одиницю виробу, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на 1000 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року – на 1000 шт., протягом другого року – на 1500 шт., протягом третього року на 1800 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

$$\Delta П_1 = (0*1000 + (15000 + 1000)*1000)* 0,8333* 0,35) * (1 - 0,18) = 3587499,857 \text{ грн.}$$

$$\Delta П_2 = (0*1000 + (15000 + 1000)*(1000+1500)* 0,8333* 0,35) * (1 - 0,18) = 9566666,284 \text{ грн.}$$

$$\Delta П_З = (0*1000 + (15000 + 1000)*(1000+1500+1800)* 0,8333* 0,35) * (1 - 0,18) = 16454666,008 \text{ грн.}$$

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 29608832,15 грн.

Розрахунок ефективності вкладених інвестицій та періоду їх окупності.

Розраховуємо приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП= (3587499,857/(1+0,1)^1)+(9566666,284/(1+0,1)^2)+(16454666,008/(1+0,1)^3) = 3261363,51 + 7906335,772 + 12362634,12 = 23530333,39 \text{ грн.}$$

Далі розраховують величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} * ЗВ, \quad (5.9)$$

де $k_{інв}$ — коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}$=2...5, але може бути і більшим;

ЗВ — загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн,

$$PV = 2 * 971644 = 1943287,70 \text{ грн.}$$

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід (NPV, Net Present Value) для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП – PV, \quad (5.10)$$

$$E_{абс} = 23530333,39 – 1943287,70 = 21587045,70 \text{ грн.}$$

Оскільки $r_{О\,в} \geq 0$ то вкладання коштів на виконання та впровадження результатів даної науково-дослідної (науково-технічної) роботи може бути доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $r_в$.

$$E_в = \sqrt{з\ (1 + 21587045,70/1943287,70} – 1 = 1,296$$

Визначимо мінімальну ставку дисконтування.

Так як $E_в > \tau_{min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Оскільки $\sigma_{т}$ < 3-х років, а саме термін окупності рівний 0,77 роки, то фінансування даної наукової розробки є доцільним.

Висновки до розділу: економічна частина даної роботи містить розрахунок витрат на розробку нового програмного продукту, сума яких складає 971644 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є висококонкурентоспроможним. Період окупності складе близько 0,77 роки.

ВИСНОВКИ

У магістерській кваліфікаційній роботі було розглянуто та проаналізовано методи адаптації Web-додатків для використання людьми з обмеженими можливостями.

У першому розділі були розглянуті статистичні дані про працевлаштування та середній освітній рівень людей із обмеженими можливостямита людей без вад. Доведено актуальність та важливість проблеми користування web-ресурсами усіх людей. Розглянуто найпопулярніші проблеми людей з певними вадами у Інтернеті та причи виникнення цих проблем . Також у розділі були приведені статистичні дані про типові помилки та проблеми сучасних сайтах по всьому світі за параметром доступності.

У другому розділі було розглянуто та проаналізовано методи та хороші практити для покращення доступності сайтів. Здійснено огляд відомих та популярних сайтів за параметром доступності з допомогою інструменту Ligthhouse і представлено можливі вирішення виявлених проблем.

У третьому розділі магістерської кваліфікаційної роботи здійснено порівняння сайтів та web-додатків, виявлено основні переваги та недоліки. Оглянуто сучасні технології для розробки клієнтської та серверної частин web-додатків та обрано необхідний стек технологій. Приведено розробку веб-застосунку з використанням методів покращення достуності сайтів.

У четвертому розділі магістерської кваліфікаційної роботи було проведено тестування розробленого продукту за параметром доступності. Розроблено інструкцію для користувача.

В п’ятому розділі роботи проведено дослідження економічної частини продукту. Кінцевий продукт є дешевшим за аналог і є висококонкурентоспроможним. Через фактор хорошого рівня доступності, забезпечено покриття більшого відстотку клієнтів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Статистика студентів з інвалідністю [Електронний ресурс] – Режим доступу: https://nces.ed.gov/fastfacts/display.asp?id=60

2. Показники працевлаштованості за освітніми досягненнями [Електронний ресурс] – Режим доступу: https://nces.ed.gov/programs/coe/pdf/coe_tad.pdf

3. Вступ до web-доступності. [Електронний ресурс] – Режим доступу: https://www.w3.org/WAI/fundamentals/accessibility-intro/

4. Лаура Калбаг. Accessebility for everyone. A Book Apart, 2017. 44 с.

5. Чому доступність це важливо? [Електронний ресурс] – Режим доступу: https://accessibility.uncg.edu/why-web-accessibility-is-important/

6. The Burden of Vision Loss [Електронний ресурс] – Режим доступу: https://www.cdc.gov/visionhealth/risk/burden.htm

7. Цифрова доступність для епілептиків [Електронний ресурс] – Режим доступу: https://speechify.com/blog/digital-accessibility-for-epilepsy/?landing_url=https%3A%2F%2Fspeechify.com%2Fblog%2Fdigital-accessibility-for-epilepsy%2F

8. Фоточутливість і судоми [Електронний ресурс] – Режим доступу: https://www.epilepsy.com/what-is-epilepsy/seizure-triggers/photosensitivity#Examples-of-Triggers

9. 53 Web Accessibility Statistics Електронний ресурс] – Режим доступу: https://ddiy.co/web-accessibility-statistics/

10. How to choose a font for accessibility [Електронний ресурс] – Режим доступу: https://www.siteimprove.com/glossary/accessible-fonts/

11. A guide to color accessibility in product design. [Електронний ресурс] – Режим доступу: https://www.invisionapp.com/inside-design/color-accessibility-product-design/

12. 6 Tips & Best Practices to Achieve Accessibility Web Design [Електронний ресурс] – Режим доступу: https://userway.org/blog/6-best-practices-in-accessibility-web-design/

13. What Is Google Lighthouse and How to Use It? [Електронний ресурс] – Режим доступу: https://www.elegantthemes.com/blog/wordpress/what-is-google-lighthouse-and-how-to-use-it?utm_source=Blog&utm_medium=Manual%20WordPress%20Targets&utm_campaign=Google%20Search&retargeting=off&gclid=CjwKCAiAheacBhB8EiwAItVO2w4FSDDcAJVqr46QO2v-vPbLtUoLY0FuuP5MnuKrtnqFy09INULULhoCMTwQAvD_BwE

14. Difference between Website and Web Application. [Електронний ресурс] – Режим доступу:https://www.guru99.com/difference-web-application-website.html

15. Comparison between Angular, React, and Vue. [Електронний ресурс] – Режим доступу: https://www.javatpoint.com/angular-vs-react-vs-vue

16. Node.js Frameworks Comparison for Your Back-end Solution. [Електронний ресурс] – Режим доступу: https://www.altexsoft.com/blog/engineering/node-js-frameworks-comparison-for-your-back-end-solution-express-js-meteor-js-sails-js-and-more/

17. 5 Quick Ways to Self-check the Accessibility of a Website. [Електронний ресурс] – Режим доступу: https://www.boia.org/blog/5-quick-ways-to-self-check-the-accessibility-of-a-website

18. W3C [Електронний ресурс] – Режим доступу: https://www.w3.org/standards/webdesign/accessibility

19. Introduction to web accessibility [Електронний ресурс] – Режим доступу: https://www.w3.org/WAI/fundamentals/accessibility-intro/

20. What is accessibility [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Learn/Accessibility/What_is_accessibility

ДОДАТОК А

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
проф., д.т.н..
_____Азаров О.Д.

"__10__" __10__2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання магістерської кваліфікаційної роботи
"Комп'ютерна система моніторингу регіонального радіомовлення"
08-23. МКР.004.00.000.ТЗ

Науковий керівник: проф.
_____ Захарченко С.М.

Студент групи 1КІ-21м
_____ Козяр С.О.

ВНТУ 2022

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР)

Підставою для виконання МКР є розкриття поточної проблеми доступності сайтів для усіх людей та аналіз сучасних методів адатації сайтів для людей з обмеженими можливостями. Актуальність роботи полягає в розробці веб-застосунку у якому буде високий рівень доступності для людей з обмеженими можливостями.

Наказ про затвердження теми дипломної роботи від «15» _вересня_ 2022 року № 205-А

2 Мета МКР і призначення розробки

2.1 Мета проекту — огляд сучасного стану достуності сайтів у всьому світі. Аналіз популярних сайтів та виведеня типових помилок при роботі з достуністю під час розробки web-продуктів. Аналіз способів покращення доступності сайтів та їх практичне використання.

2.2 Призначення розробки — створення веб-застосунку з використанням методів покращення доступності;

3 Вихідні дані для виконання МКР

3.1 Розкриття актуальності роботи над доступністю та важливість аспекту доступності для усіх людей;

3.2 Аналіз усіх можливих проблем сайтів за параметром доступності та огляд методів вирішення цих проблем;

3.3 Розробити web-додаток з використанням розглянутих методів;

3.4 Виконання розрахунків для доведення доцільності нової розробки з економічної точки зору;

3.5 Протестувати додаток за параметром достуності порівняти із аналогами.

4 Вимоги до виконання МКР

Головна вимога — використати, методи покращення доступності сайтів для досягнення максимального результату за параметром доступності

5 Етапи МКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

Таблиця А.1 — Етапи МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Огляд проблем з якими стикаються люди з обмеженими можливостями в Інтернеті.	24.09.2022	27.09.2022	Розділ 1
2	Огляд сучасного стану достуності у світі.	28.09.2022	29.09.2022	Розділ 1
3	Огляд методів для покращення доступності сайтів. Опис способів застосування методів для вирішення проблем.	15.10.2022	18.10.2022	Розділ 2
4	Розробка додатку із використанням методів окращення доступності.	23.10.2022	05.11.2022	Розділ 3
5	Тестування додатку за параметром доступності.	06.11.2022	08.11.2022	Розділ 4
6	Підготовка економічної частини	10.11.2022	15.11.2022	Розділ 5
7	Апробація та впровадження результатів дослідження	18.11.2022	21.11.2022	Тези доповідей
8	Опублікування результатів досліджень	25.11.2022	28.11.2022	Стаття
9	Оформлення пояснювальної записки, графічного матеріалу і презентації	29.11.2022	15.12.2022	ПЗ, графіч. матеріал і презентація
10	Підготовка супроводжуючих документів, їх підписування, проходження нормоконтролю та тесту на плагіат	17.12.2022	19.12.2022	Оформлені документи

.

6 Матеріали, що подаються до захисту МКР

До захисту подаються: пояснювальна записка МКР, протокол попереднього захисту МКР на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних екзаменів, анотації до МКР українською та іноземною мовами, довідка про відповідність оформлення МКР діючим вимогам.

7 Порядок контролю виконання та захисту МКР

Виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання МКР

8.1 При оформлювання МКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— міждержавний ГОСТ 2.104-2006 «Едина система конструкторської документації. Основні написи»;

— Методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія». Кафедра обчислювальної техніки ВНТУ 2022;

— документами на які посилаються у вище вказаних.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21».

ДОДАТОК Б

Вигляд домашньої сторінки додатку

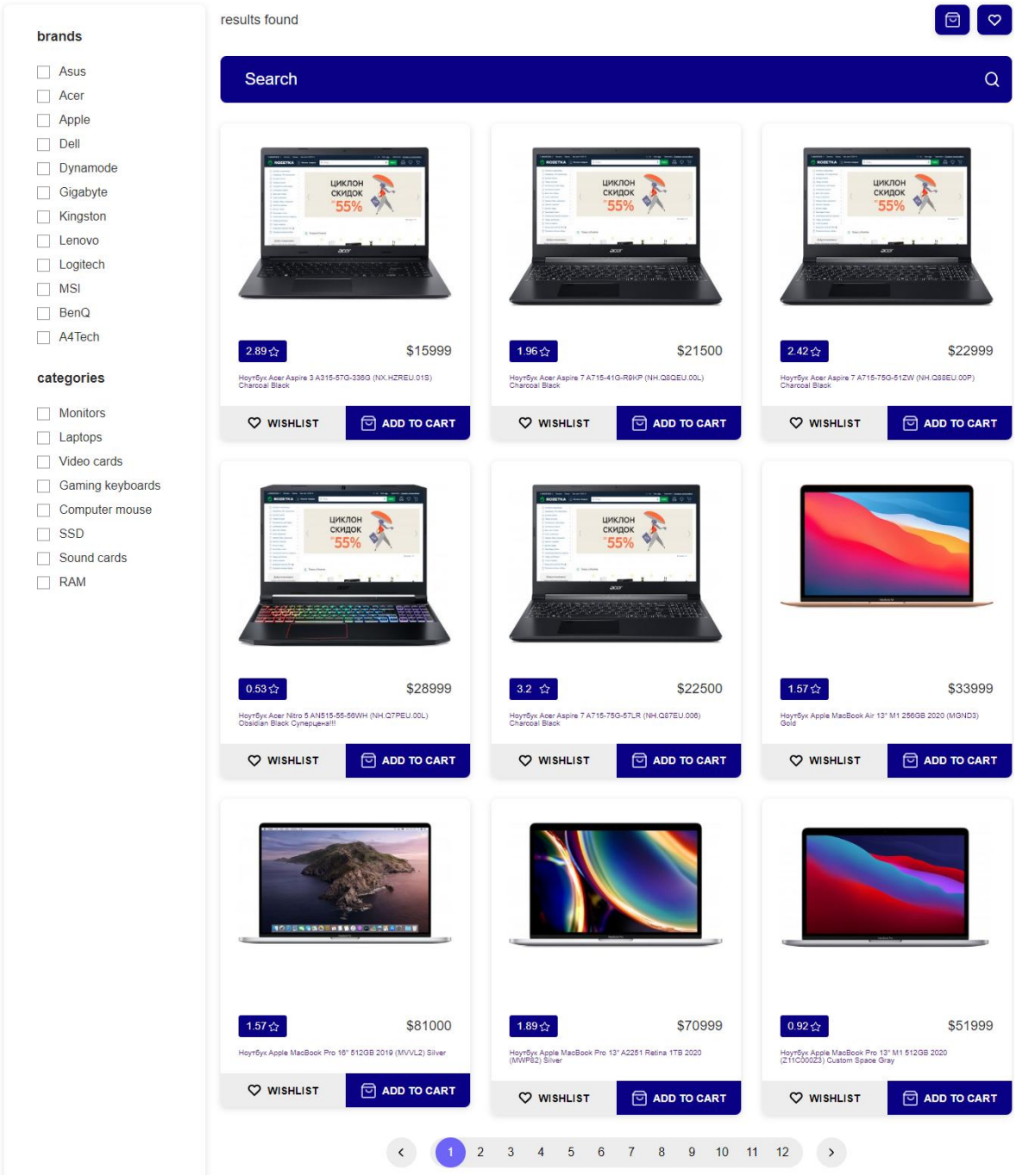

Рисунок Б.1 — Домашня сторінка додатку

ДОДАТОК В

Лістнг коду створених компонентів

CardComponent class

```
import { Component, Input } from '@angular/core';

@Component({
  selector: 'app-card',
  templateUrl: './card.component.html',
  styleUrls: ['./card.component.scss'],
})
export class CardComponent {
  @Input() card = {
    id: '',
    title: '',
    rating: 0,
    price: 0,
    category: '',
    images: [''],
    brand: '',
    isCart: false,
    isWished: false,
    uniqId: 0,
  };

  constructor(
  ) {}
}
```

CardListComponent class

```
import { Component, Input, OnInit } from '@angular/core';

@Component({
  selector: 'app-cardlist',
  templateUrl: './cardlist.component.html',
  styleUrls: ['./cardlist.component.scss']
})
export class CardlistComponent {

  @Input() currentPageProducts = [];

}
```

CardListContainerComponent class

```
import { Component, OnInit } from '@angular/core';
import { ProductStorageService } from '../product-storage.service';
import { Subscription, BehaviorSubject, take } from "rxjs"

@Component({
  selector: 'app-cardlist-container',
  templateUrl: './cardlist-container.component.html',
  styleUrls: ['./cardlist-container.component.scss']
})
export class CardlistContainerComponent implements OnInit {

  filtersState: any = {
    q: '',
    categories: '',
    brands: '',
    page: 1,
    limit: 9,
  }

  arrayForPagination: Array<number> = [];

  filters = new BehaviorSubject(this.filtersState);
  filtersSubscription$: Subscription = new Subscription();
  currentPageProducts = [];

  constructor(private productService: ProductStorageService){}

  ngOnInit(): void {
    this.filtersSubscription$ = this.filters.subscribe(state => {
      this.productService.getFilteredProducts(state).pipe(
        take(1)
      ).subscribe(response => {
        console.log(response)
        const numberOfPages = Math.ceil(response.numberOfProducts / 9);
        this.arrayForPagination = Array.from(Array(numberOfPages).keys());
        this.currentPageProducts = response.products;
      })
    })
  }

  ngOnDestroy() {
    this.filtersSubscription$.unsubscribe();
  }
```

```
  onCheck(event: any): void {
    if(event) {
      this.filtersState[event.type] = event.value;
    }
    this.filters.next(this.filtersState);
  }

  onSearch(event: string): void {
    this.filtersState.q = event;
    this.filters.next(this.filtersState);
  }

  onSelectPage(event: number) {
    this.filtersState.page = event;
    this.filters.next(this.filtersState);
  }

}
```

FilterFormComponent class

```
import { Component, Output, Input, EventEmitter } from '@angular/core';

@Component({
  selector: 'app-filter-form',
  templateUrl: './filter-form.component.html',
  styleUrls: ['./filter-form.component.scss']
})
export class FilterFormComponent {

  @Output() filtersChecked: EventEmitter<any> = new EventEmitter();
  filtersKeys: any = [];
  @Input() filters: any = {brands: [], categories: []}

  ngOnInit(): void {
    this.filtersKeys = Object.keys(this.filters);
  }

  onCheck(input: any):void {
    const type = this.getType(input.value);
    this.uncheckOtherInputs(type, input.value);
    this.filtersChecked.emit({
      value: !input.checked ? input.value.toLowerCase().replace(/ /g, '_') : '',
      type
    });
```

```
  }

  getType(value: string): string {
    const filtersTypes = Object.keys(this.filters);
    return filtersTypes.filter(key =>
      this.filters[key].find((item: any) => item.value === value)
    )[0];
  }

  uncheckOtherInputs(type: string, value: string){
    this.filters[type] =  this.filters[type].map((item: any)  =>
      item.value !== value ?
       {...item, checked: false} :
       {...item, checked: !item.checked}
    );
  }

}
```

PaginationComponent class

```
import { Component, EventEmitter, Input, Output } from '@angular/core';

@Component({
  selector: 'app-pagination',
  templateUrl: './pagination.component.html',
  styleUrls: ['./pagination.component.scss'],
})
export class PaginationComponent {
  @Input() arrayForPagination: Array<number> = [];
  @Output() selectedPage = new EventEmitter<number>()
  currentPage = 0;

  selectPage(event: Event, numberOfPage: number) {
    this.currentPage = numberOfPage;
    event.preventDefault();
    this.selectedPage.emit(numberOfPage + 1);
  }

}
```

SearchComponent class

```
import { Component, Output, EventEmitter } from '@angular/core';
import { ProductStorageService } from '../product-storage.service';
```

```
import { SearchService } from '../search.service';

@Component({
  selector: 'app-search',
  templateUrl: './search.component.html',
  styleUrls: ['./search.component.scss']
})
export class SearchComponent {

  @Output() value: EventEmitter<any> = new EventEmitter();

  constructor() { }


  getSearhUrl(event:Event, searchText:string):void {
    event.preventDefault();
    this.value.emit(searchText);
  }

}
```

SidebarComponent class

```
import { Component, EventEmitter, OnInit, Output } from '@angular/core';
import { FiltersService } from '../filters.service';

@Component({
  selector: 'app-sidebar',
  templateUrl: './sidebar.component.html',
  styleUrls: ['./sidebar.component.scss']
})
export class SidebarComponent implements OnInit {

  @Output() checkedFilters: EventEmitter<any> = new EventEmitter();

  filters: any = {brands: [], categories: []}

  constructor(private filtersService: FiltersService) { }

  ngOnInit(): void {
    this.filtersService.getCategories().subscribe(categories => {
      this.filters.categories = categories.map((item: string) => ({ value: item, checked:
false}));
    });
    this.filtersService.getBrands().subscribe(brands => {
```

```
      this.filters.brands = brands.map((item: string) => ({ value: item, checked:
false}));
    });
  }

  onCheck(event: Array<string>):void {
    this.checkedFilters.emit(event);
  }

}
```

ДОДАТОК Г

Лістинг коду html шаблонів

```
<article class="products__item product-card product-card--with-rating" >
   <div class="product-card__main">
     <div class="product-card__img">
       <img [src]="card.images[0]" alt="product" />
     </div>
     <div class="product-card__info">
       <div class="product-card__wr">
         <div class="product-card__rating">
           <span>
             {{ card.rating }}
           </span>
           <img src="../../../assets/img/icons/star.svg" alt="rating-star" />
         </div>
         <div tabindex="0" class="product-card__price">${{ card.price }}</div>
       </div>

       <a tabindex="0" href="/product" aria-label="product name"
         class="product-card__description"
       >
           {{ card.title }}
       </a>
     </div>
   </div>
   <div class="product-card__btns">
     <button  tabindex="0" aria-label="add to wishlist" class="product-card__btn
btn">
       <img
         src="../../assets/img/icons/heart.svg"
         alt="heart"
       />
       <span>WISHLIST</span>
     </button>
     <button  tabindex="0"
       aria-label="add to cart"
       class="product-card__btn btn btn--violet"
     >
       <img
         src="../../../assets/img/icons/bag.svg"
         alt="bag"
       />
```

```
        <span>
          ADD TO CART
        </span>
      </button>
    </div>
  </article>

<ul class="products__inner">
    <li *ngFor="let card of currentPageProducts">
      <app-card [card]="card"></app-card>
    </li>
</ul>

<div class="container cardlist-container">
<app-sidebar (checkedFilters)="onCheck($event)"></app-sidebar>
<div>
  <div class="products__search-res">
    <p> results found</p>
    <div class="products__btns">
      <a routerLink="/cart" class="btn btn--only-img btn--violet">
        <img src="../../../assets/img/icons/bag.svg" alt="heart" />
      </a>

      <a routerLink="/wishlist" class="btn btn--only-img btn--violet">
        <img
          src="../../../assets/img/icons/white-heart.svg"
          alt="heart"
        />
      </a>
    </div>
  </div>
    <app-search (value)="onSearch($event)"></app-search>
    <app-cardlist [currentPageProducts]="currentPageProducts"></app-cardlist>
    <app-pagination (selectedPage)="onSelectPage($event)"
[arrayForPagination]="arrayForPagination"></app-pagination>
</div>

</div>

<main>
  <section class="products-list">
    <div class="container">
      <a
        routerLink="/"
```

```
      class="btn btn--violet btn--large clear-wishlist btn--border-radius"
      >back to home</a
    >
    <a
    routerLink="/wishlist"
    class="btn btn--violet btn--large clear-wishlist btn--border-radius"
    >go to wishlist</a
  >
  </div>
 </section>
</main>



<form class="sidebar__form"  >
 <div *ngFor="let item of filtersKeys;">
 <div class="sidebar__filter" data-element="body">
  <h3 tabindex="0">
   <strong> {{item}} </strong>
  </h3>
  <label
   class="field field--checkbox sidebar__field"

    *ngFor="let input of filters[item];"
    (click)="onCheck(input)"
  >
   <input
   tabindex="0"
    #input
    type="checkbox"
    [value]="input.value"
    [checked]="input.checked"
   />
   <span class="field__checkbox-substitute"> </span>
   <span class="field__info">
    <span class="field__name">
     {{input.value}}
    </span>
   </span>
  </label>
 </div>
 </div>
</form>
```

```
<app-cardlist-container></app-cardlist-container>
<nav class="products-list__pagination pagination">
   <button
    class="pagination__btn pagination__btn--left"
   >
    <img src="../../../assets/img/icons/arrow-left.svg" alt="prev" />
   </button>
   <ul class="pagination__list">
    <li class="pagination__item"
       *ngFor="let item of arrayForPagination;"
    >
     <a
       aria-label="page"
       [href]="'https://store/products/page/' + (item + 1)"
       (click)="selectPage($event, item)"
       role="button"
       [ngClass]="{
        'pagination__link--current': currentPage === item
       }"
     >
     {{item + 1}}
     </a>
    </li>
   </ul>
   <button
    class="pagination__btn pagination__btn--right"
   >
    <img src="../../../assets/img/icons/arrow-left.svg" alt="prev" />
   </button>
  </nav>
<form
  class="products__search search"
  (submit)="getSearhUrl($event, input.value)"
>
  <input type="text" placeholder="Search" #input tabindex="0" />
  <button aria-label="submit for search" type="submit" tabindex="0">
    <img src="../../../assets/img/icons/search.svg" alt="search" />
  </button>
</form>
<app-filter-form [filters]="filters" (filtersChecked)="onCheck($event)"></app-filter-
form>
```

ДОДАТОК Д

Лістинг стилів компонентів

```
.product-card {
    display: flex;
    flex-direction: column;
    justify-content: space-between;
    width: 100%;
    border-radius: 8px;
    box-shadow: 0px 2px 8px 0px rgba(00, 00, 00, 0.1352);
    background-color: #fff;
    // .product-card__main
    &__main {
      padding: 3.1rem 2.49rem 2.2rem 2.372rem;
      box-sizing: border-box;
      display: flex;
      width: 100%;
      flex-direction: column;
      justify-content: space-between;
      height: 100%;
    }

    // .product-card__price
    &__price {
      color: #2c2c2c;
      font-size: 1.8rem;
      line-height: 2.1rem;
    }

    // .product-card__btn
```

```
&__btn {
  &:first-of-type {
    border-radius: 0 0 0 8px;
    @media screen and (max-width: 1200px) {
      border-radius: 0;
    }
  }

  &:last-of-type {
    border-radius: 0 0 8px 0;
    @media screen and (max-width: 1200px) {
      border-radius: 0 0 8px 8px;
    }
  }
}

// .product-card__btns
&__btns {
  display: grid;
  grid-template-columns: repeat(2, 1fr);
  @media screen and (max-width: 1200px) {
    grid-template-columns: 100%;
  }
}

// .product-card__description
&__description {
  h3 {
    color: #2c2c2c;
    line-height: 1.9rem;
```

```
    font-size: 1.6rem;
    margin-bottom: 0.3rem;
    font-weight: 400;
  }
  p {
    font-weight: 300;
    font-size: 1.2rem;
    line-height: 1.4rem;
    color: #2c2c2c;
  }
}

// .product-card__rating
&__rating {
  padding: 0.6rem 1rem;
  display: flex;
  justify-content: space-between;
  align-items: center;
  min-width: 6.4rem;
  border-radius: 3px;
  box-sizing: border-box;
  background-color: #090086;

  span {
    color: #fff;
    font-size: 1.4rem;
    line-height: 1.7rem;
  }
}
```

```
// .product-card__wr
&__wr {
  display: flex;
  align-items: center;
  margin-bottom: 1.6rem;
  justify-content: flex-end;
}

// .product-card__img
&__img {
  margin-bottom: 3.2rem;
  display: flex;
  position: relative;
  width: 100%;
  padding-bottom: 79.5%;
  justify-content: center;

  img {
    position: absolute;
    left: 50%;
    min-width: 50%;
    max-height: 100%;
    max-width: 100%;
    top: 0;
    transform: translateX(-50%);
    object-fit: contain;
  }
}

// .product-card--with-rating
```

```
    &--with-rating {
      .product-card__wr {
        justify-content: space-between;
      }
    }
  }

:host {
    .products {
      // .products__inner
      &__inner {
        display: grid;
        gap: 2.8rem;
        grid-template-columns: repeat(3, 1fr);
        @media screen and (max-width: 1200px) {
          grid-template-columns: repeat(2, 1fr);
          gap: 1.6rem;
        }

        @media screen and (max-width: 720px) {
          grid-template-columns: 100%;
        }
      }
    }
    .products-list {
      // .products-list__bread-cramps
      &__bread-cramps {
        margin-bottom: 4.4rem;
      }
      // .products-list__pagination
```

```
      &__pagination {
        margin: 4rem auto 0 auto;
      }
      // .products-list__inner
      &__inner {
        display: grid;
        grid-template-columns: 29% 69%;
        gap: 2.5rem;

        @media screen and (max-width: 1200px) {
          gap: 1.6rem;
        }
        @media screen and (max-width: 720px) {
          grid-template-columns: 100%;
        }
      }
    }
  }


.cardlist-container {
  margin: 20px;
  padding-left: 0;
  display: grid;
  grid-template-columns: 20% 1fr;
  gap: 2rem;
}
.products__search-res {
  display: flex;
  align-items: center;
```

```
  justify-content: space-between;
  margin-bottom: 2.8rem;
  p {
    font-size: 1.8rem;
    line-height: 2.1rem;
    color: #2c2c2c;
  }
}
.products__btns {
  display: flex;
  a {
    margin-right: 10px;
    &:last-of-type {
      margin-right: 0;
    }
  }
}


.field {
    position: relative;
    display: flex;
    align-items: center;
    width: 100%;
    cursor: pointer;
    input[type="radio"],
    input[type="checkbox"] {
      position: absolute;
      left: 0;
      top: 0;
```

```
  width: 0;
  height: 0;
  opacity: 0;
}

input:checked + .field__radio-substitute {
  border: 1px solid #090086;
}

input:checked + .field__radio-substitute,
input:checked + .field__checkbox-substitute {
  background-color: #090086;
}
// .field__info
&__info {
  display: flex;
  align-items: center;
  justify-content: space-between;
  width: 100%;
}
// .field__name
&__name {
  margin-left: 1.2rem;
  font-size: 1.6rem;
  line-height: 2;
  font-weight: 300;
  color: #2c2c2c;
}

// .field--checkbox
```

```
  &--checkbox {
    .field__name {
      margin-right: 0.8rem;
    }
  }

  //  .field__value
  &__value {
    font-size: 1.4rem;
    line-height: 3.2rem;
    color: #2c2c2c;
  }
  //.field__checkbox-substitute
  //.field__radio-substitute
  &__checkbox-substitute,
  &__radio-substitute {
    display: block;
    width: 1.7rem;
    height: 1.5rem;
    border: 1px solid #979797;
    background-color: transparent;
  }

  // .field__radio-substitute
  &__radio-substitute {
    border-radius: 50%;
  }
}
.sidebar__form {
  height: 100%;
```

```
    background-color: #fff;
    border-radius: 8px;
    padding: 3.2rem 4.4rem;
    margin-bottom: 4rem;
    box-shadow: 0px 2px 8px 0px rgba(00, 00, 00, 0.1352);
    @media screen and (max-width: 1200px) {
      padding: 2rem;
    }
  }
  .sidebar__filter {
    padding-top: 3.6rem;
    border-top: 1px solid #d6d6d6;
    margin-bottom: 2.8rem;

    h3 {
      color: #2c2c2c;
      margin-bottom: 2rem;
    }

    &:first-of-type {
      padding-top: 0;
      border-top: none;
    }
  }


.pagination {
  display: flex;
  align-items: center;
  justify-content: center;
```

```
@media screen and (max-width: 720px) {
  width: 100%;
  justify-content: space-between;
  flex-wrap: wrap;
}
// .pagination__btn
&__btn {
  width: 4rem;
  display: flex;
  align-items: center;
  justify-content: center;
  height: 4rem;
  border-radius: 50%;
  background-color: #ededed;
  @media screen and (max-width: 720px) {
    width: 2.8rem;
    height: 2.8rem;
  }

  img {
    width: 0.7rem;
    height: 1.2rem;
  }
}

// .pagination__btn--right
&__btn--right {
  img {
    transform: rotate(180deg);
  }
```

```
  }

  // .pagination__list
  &__list {
    margin: 0 1.8rem;
    padding: 0 1.2rem;
    display: flex;
    align-items: center;
    height: 4rem;
    border-radius: 20px;
    counter-reset: page;
    background-color: #ededed;
    @media screen and (max-width: 720px) {
      height: 2.8rem;
      margin: 0 0.4rem;
    }
  }
}
.pagination__item {
  width: 3rem;
  height: 100%;
  margin-right: 1rem;
  @media screen and (max-width: 720px) {
    width: 1.2rem;
  }

  &:last-of-type {
    margin-right: 0;
  }
}
```

```
li a {
  position: relative;
  width: 100%;
  height: 100%;
  display: block;
  font-size: 1.6rem;color: #000;
  display: flex;
  align-items: center;
  justify-content: center;

  &:after,
  &:before {
    content: "";
    position: absolute;
    left: 50%;
    top: 50%;
    transform: translate(-50%, -50%);
  }

  &.pagination__link--current {
    position: relative;
    z-index: 5;
    color: #fff;

    &:after {
      border-radius: 50%;
      width: 4rem;
      height: 4rem;
      background-color: #6f64f8;
```

```
      z-index: -1;
      @media screen and (max-width: 720px) {
        width: 2.8rem;
        height: 2.8rem;
      }
    }

    &:before {
      color: #fff;
    }
  }
}


.search {
    position: relative;
    margin-bottom: 2.8rem;
    button {
      position: absolute;
      top: 50%;
      transform: translateY(-50%);
      right: 2rem;
      width: 1.6rem;
      height: 2rem;
    }
    input {
      box-sizing: border-box;
      padding: 1.7rem 3.8rem 1.7rem 3.2rem;
      width: 100%;
      height: 6.2rem;
```

```
    background-color: #090086;
    border-radius: 8px;
    &::placeholder {
      color: #fff;
    }
  }
}
```

ДОДАТОК Е

Лістинг коду сервісу ProductStorageService

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpParams } from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class ProductStorageService {


  constructor(private httpClient: HttpClient) { }

  getNumberOfAllProducts(): Observable<any> {
    return this.httpClient.get('http://localhost:5000/products/number');
  }

  getFilteredProducts(paramsInfo: string): Observable<any> {
    let params = new HttpParams();
    Object.entries(paramsInfo).map(([key, value]) => {
      params = params.append(key, value)
    })a
    return this.httpClient.get('http://localhost:5000/search', { params: params });
  }

}
```

ДОДАТОК Ж

Протокол перевірки кваліфікаційної роботи

Назва роботи: Методи адаптації Web-додатків для використання людьми з обмеженими можливостями

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 97.8% Схожість 2.2%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.

☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Козяр С.О.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Захарченко С.М.
(підпис) (прізвище, ініціали)