МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
КРИВОРІЗЬКИЙ ФАХОВИЙ КОЛЕДЖ
ДЕРЖАВНОГО НЕКОМЕРЦІЙНОГО ПІДПРИЄМСТВА
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»
Циклова комісія *комп'ютерних систем та мереж*
(повна назва циклової комісії)

Допустити до захисту
Голова випускової циклової комісії
комп'ютерних систем та мереж
(повна назва циклової комісії)
_____ *Ірина КРАВЧУК*
(підпис) (ім'я, ПРІЗВИЩЕ)
« *10* » *06* 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬО-ПРОФЕСІЙНОГО СТУПЕНЯ
ФАХОВИЙ МОЛОДШИЙ БАКАЛАВР

Тема: *"Інформаційний веб-ресурс для перегляду відеоновинок"*

Група: *3-013* Спеціальність: *123 «Комп'ютерна інженерія»*

Здобувач освіти _____ *Ростислав Скрипник*
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ *Олександр ГРИНЧЕНКО*
(підпис) (ім'я, ПРІЗВИЩЕ)

Консультант з оформлення
пояснювальної записки _____ *Оксана ОСАДЧА*
(підпис) (ім'я, ПРІЗВИЩЕ)

Кривий Ріг 2025 р.

КРИВОРІЗЬКИЙ ФАХОВИЙ КОЛЕДЖ
ДЕРЖАВНОГО НЕКОМЕРЦІЙНОГО ПІДПРИЄМСТВА
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

Відділення *комп'ютерної та програмної інженерії*
Циклова комісія *комп'ютерних систем та мереж*
Освітньо-професійний ступінь *фаховий молодший бакалавр*
Спеціальність *123 «Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Голова випускової циклової комісії
комп'ютерних систем та мереж
(повна назва циклової комісії)
_____ *Ірина КРАВЧУК*
(підпис) (ім'я, ПРІЗВИЩЕ)
« *01* » *05* 2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ОСВІТИ

Скрипнику Ростиславу Сергійовичу
(прізвище, ім'я, по батькові)

1.Тема роботи *"Інформаційний веб-ресурс для перегляду відеоновинок"*

Керівник роботи *Гринченко Олександр Сергійович, викладач вищої категорії*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по коледжу від « *04* » *04* 2025 року № *50-ст*

2. Строк подання здобувачем освіти роботи з _____ по _____

3. Вихідні дані до роботи *ПЗ для створення локального сервера, браузер для перевірки сайту та заходу на нього*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно

РОЗДІЛ 1: АНАЛІЗ ОБЛАСТІ ФУНКЦІОНАЛУ ТА ІМПОРТУВАННЯ ВІДЕО У

РОЗДІЛ 2: ПРОЕКТУВАННЯ СИСТЕМИ ВЕБ-ЗАСТОСУНКУ

РОЗДІЛ 3: СТВОРЕННЯ ВЕБ-ЗАСТОСУНКУ

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Презентація Microsoft PowerPoint

6. Консультанти розділів роботи (проекту)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	*Узгодження завдання з керівником дипломної роботи*	*04.04.2025-07.04.2025*	*Виконано*
2	*Вивчення літератури за темою дипломної роботи*	*08.04.2025-14.04.2025*	*Виконано*
3	*Розділ 1: Аналіз області функціоналу та імпортування відео у веб-застосунок*	*15.04.2025-21.04.2025*	*Виконано*
4	*Розділ 2: Проектування веб-застосунку*	*22.04.2025-28.04.2025*	*Виконано*
5	*Розділ 3: Створення веб-застосунку*	*29.04.2025-02.05.2025*	*Виконано*
6	*Написання пояснювальної записки*	*12.05.2025-23.05.2025*	*Виконано*
7	*Підготовка презентації*	*26.05.2025-30.05.2025*	*Виконано*
8	*Попередній захист кваліфікаційної роботи*	*02.06.2025-06.06.2025*	*Виконано*
9	*Захист дипломної роботи*		

Здобувач освіти _____ *Ростислав Скрипник*
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ *Олександр ГРИНЧЕНКО*
(підпис) (ім'я, ПРІЗВИЩЕ)

StrikePlagiarism.com

Дата звіту 6/13/2025

Дата редагування —

Документ прийнятий

Звіт подібності

метадані

Назва організації
Ukrainian national aviation university

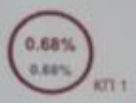
Заголовок
Скрипник Р_3-013_2025_КПІ

Автор Науковий керівник / Експерт
СкрипникГринченко О.

підрозділ
Криворізький Фаховий коледж

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високе значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1

КЦ

25	9643	72141
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	R	0
Інтервали	A	0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)	a	4

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку 4 джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз

Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	ІП3-31_Михайловський_В.В. [redacted] 12/10/2024 Lutsk National Technical University course papers (Lutsk National Technical University course papers)	12 0.12 %
2	Думітрович Євген_група ПЗ-41 [redacted] 6/5/2024 Chernivtsi Industries Applied College (Chernivtsi Industries Applied College)	11 0.11 %

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Інформаційний веб-ресурс для перегляду відеоновинок» містить: 60 сторінок, 52 рисунки, 15 використаних джерел

VISUAL STUDIO CODE, ФАЙЛИ *PHP, JS, HTML & JSON*

Метою даної роботи є створення інтерактивного вебзастосунку, що дозволяє користувачам переглядати трейлери нових фільмів або серіалів, знайомитись з їх описами та швидко орієнтуватись у сучасних відео-тенденціях. Такий ресурс виконує інформаційно-ознайомчу функцію і орієнтований на широку аудиторію.

У ході реалізації було застосовано сучасні вебтехнології: *HTML* та *CSS* для створення структури й стилізації сторінок, *JavaScript* для реалізації клієнтської логіки, а також *PHP* з *JSON*-файлами для обробки користувацьких даних, реєстрації, авторизації та взаємодії із збереженою інформацією.

ЗМІСТ

СПИСОК СКОРОЧЕНЬ ТА ТЕРМІНІВ

HTML — HyperText Markup Language (Мова гіпертекстової розмітки)

CSS — Cascading Style Sheets (Каскадні таблиці стилів)

JS — JavaScript (Скриптова мова програмування для клієнтської частини) *PHP — Hypertext Preprocessor* (Серверна мова програмування для веб-розробки) *DB — Database* (База даних)

JSON — JavaScript Object Notation (Формат обміну даними)

SQL — Structured Query Language (Мова структурованих запитів для роботи з базами даних)

XAMPP — Cross-platform, Apache, MySQL, PHP, Perl (Програмний пакет для локального веб-серверного середовища)

CRUD — Create, Read, Update, Delete (Операції з базами даних)

ВСТУП

У сучасному світі, де інформаційні технології активно інтегруються в усі сфери життя, зростає потреба у швидкому доступі до мультимедійного контенту, зокрема до відео. Популярність відеосервісів і онлайн-платформ зумовлює актуальність розробки спеціалізованих веб-ресурсів, які дозволяють користувачам переглядати відео-новинки, орієнтуватися в сучасному кіно- та медіапросторі. Створення власного інформаційного ресурсу дає змогу задовольнити попит на локалізовані, зручні та функціональні інтерфейси для перегляду, що особливо важливо в контексті українського сегменту Інтернету. Тема є актуальною також з огляду на розвиток *frontend*- і *backend*-технологій, які дозволяють створювати

ефективні, адаптивні та безпечні вебзастосунки.

Розробка інформаційного веб-ресурсу для перегляду відео-новинок безпосередньо пов’язана з напрямами підготовки у сфері інформаційних технологій, комп’ютерної інженерії та програмної інженерії. Проєкт відповідає сучасним тенденціям цифровізації та цифрового контенту, що входять до національних та міжнародних програм розвитку *IT*-сфери, таких як цифрова трансформація суспільства, впровадження е-ресурсів та підвищення цифрової грамотності. Кваліфікаційна робота також узгоджується з освітньо-професійною програмою підготовки спеціалістів з розробки та підтримки вебсистем і сервісів, охоплюючи як розробку інтерфейсів, так і серверне програмування, взаємодію з базами даних та захист користувацьких даних.

РОЗДІЛ 1
АНАЛІЗ ОБЛАСТІ ФУНКЦІОНАЛУ ТА ІМПОРТУВАННЯ ВІДЕО У ВЕБ-ЗАСТОСУНОК

1.1 Аналіз та теоретичне ознайомлення з веб-ресурсами для імпортування відео на сайт

Є кілька веб-застосунків для перегляду та імпортування відео

наприклад: 1.*YouTube*

YouTube — це популярний відеохостинг, назва якого походить від англійських слів *you* («ти») та *tube* (жаргонне «телевізор»). Платформа з’явилася 14 лютого 2005 року завдяки трьом колишнім працівникам компанії *PayPal* — Чаду Герлі, Стівену Чену та Джаведу Каріму. Наразі сервіс є частиною компанії Google. За даними Alexa Internet, станом на серпень 2019 року *YouTube* займав друге місце серед найвідвідуваніших сайтів у світі.

Користувачі платформи можуть завантажувати власні відео, переглядати ролики інших, ставити вподобання, залишати коментарі та ділитися контентом. Простота інтерфейсу та легкість взаємодії зробили сервіс одним із наймасовіших ресурсів для обміну відеоматеріалами. Тут можна знайти як професійно зняті відео, так і аматорські записи, включно з влогами.

У жовтні 2006 року компанія *Google* придбала *YouTube* за 1,65 мільярда доларів США. За словами одного з творців платформи, Чада Герлі, її популярність

пояснюється не лише можливістю поділитися власним відео зі світом, а й тим, що сайт дозволяє легко знаходити та рекомендувати контент іншим користувачам.

До початку 2012 року щоденна кількість переглядів на сайті сягнула 4 мільярдів. На рисунку 1.1 зображено головну сторінку *YouTube.*

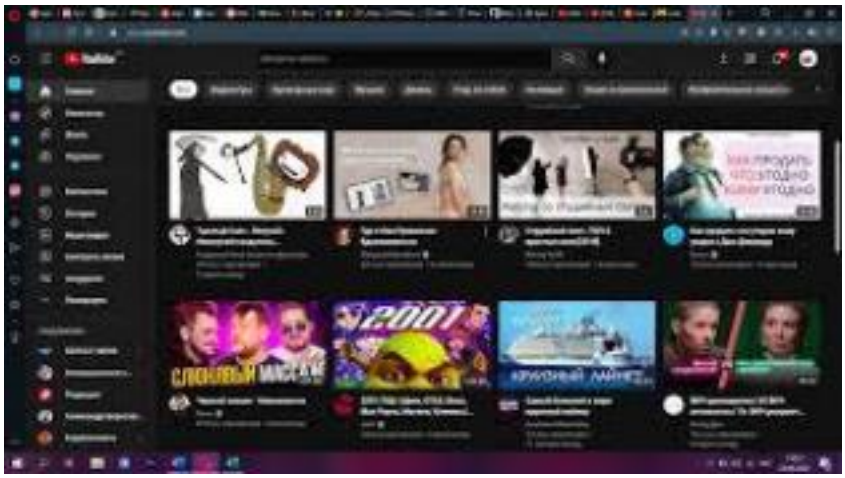


Рисунок 1.1 – Головна сторінка *YouTube*

2. *Vimeo*

Vimeo був заснований у 2004 році Заком Клейном та Джейком Лодвіком разом з групою ентузіастів кінематографії. Їхня ідея полягала в тому, щоб створити платформу, яка допоможе людям ділитися власними креативними відеопроєктами, авторськими історіями та мистецькими роботами. Назва "*Vimeo*" є поєднанням слів *video* (відео) і *me* (я), водночас це анаграма англійського слова *movie* (фільм), що символізує творчу свободу і самовираження через відео.

У жовтні 2007 року *Vimeo* став першим відеохостингом, який запровадив підтримку високої чіткості *HD* (1280×720), випередивши своїх конкурентів у сфері якості контенту. З часом сервіс почав підтримувати ще вищі роздільності, включно з 4*K* (4096×2160), що зробило його особливо привабливим для професіоналів, які прагнули демонструвати свої роботи у найкращій якості.

Станом на 2017 рік *Vimeo* вже мав понад 25 мільйонів зареєстрованих користувачів, а загальна кількість відвідувачів платформи сягала приблизно 170 мільйонів осіб. Популярність сервісу значною мірою пояснюється тим, що він орієнтується на творчу спільноту, надаючи чистий інтерфейс без реклами та високий рівень контролю над завантаженим відео.

У березні 2022 року *Vimeo* припинив реєстрацію нових користувачів з Росії та заблокував доступ до державних ЗМІ цієї країни, що були заборонені в Європейському Союзі.

На рисунку 1.2 зображено головну сторінку *Vimeo*.

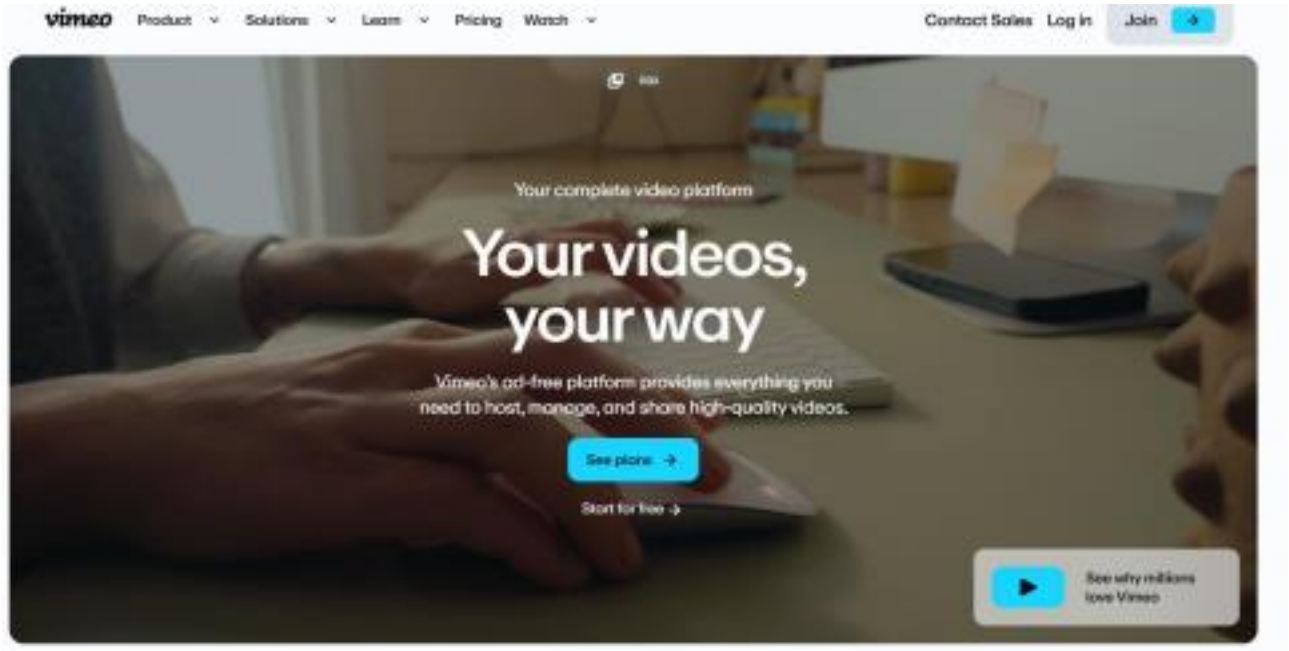


Рисунок 1.2 – Головна сторінка *Vimeo*

3. *Dailymotion*

Dailymotion — це французька відеоплатформа, яка була заснована у березні 2005 року в Парижі Бенжаміном Боденом (*Benjamin Bejbaum*) і Олів'є Пуассоном (*Olivier Poitrey*). Вона виникла як альтернатива *YouTube,* але з дещо іншим акцентом — на професійний відеоконтент, новини, культурні події та розваги. На відміну від *YouTube*, де переважає контент, створений користувачами, *Dailymotion* зосередився на співпраці з великими медіакомпаніями.

Платформа швидко здобула популярність у Європі завдяки своїй локалізованій стратегії — вона адаптується під різні країни, пропонуючи локальні мови, контент і партнерські програми. Згодом компанія розширила свою присутність у США, Азії та Південній Америці.

У 2015 році більшу частину *Dailymotion* придбала французька телекомунікаційна компанія *Orange*, а пізніше контроль над сервісом перейшов до *Vivendi* — великого медіахолдингу, якому також належать *Universal Music Group* і *Canal+*. Завдяки цьому *Dailymotion* отримав доступ до великої бібліотеки професійного контенту, включаючи музику, трейлери, телешоу, новини та документальні фільми.

Dailymotion підтримує якісне *HD*-відео, мобільні додатки для *iOS* та *Android*, а також можливість вбудовування відео на сторонні вебсайти — так само, як *YouTube* та *Vimeo*. На рисунку 1.3 зображено голвну сторінку *DailyMotion.*

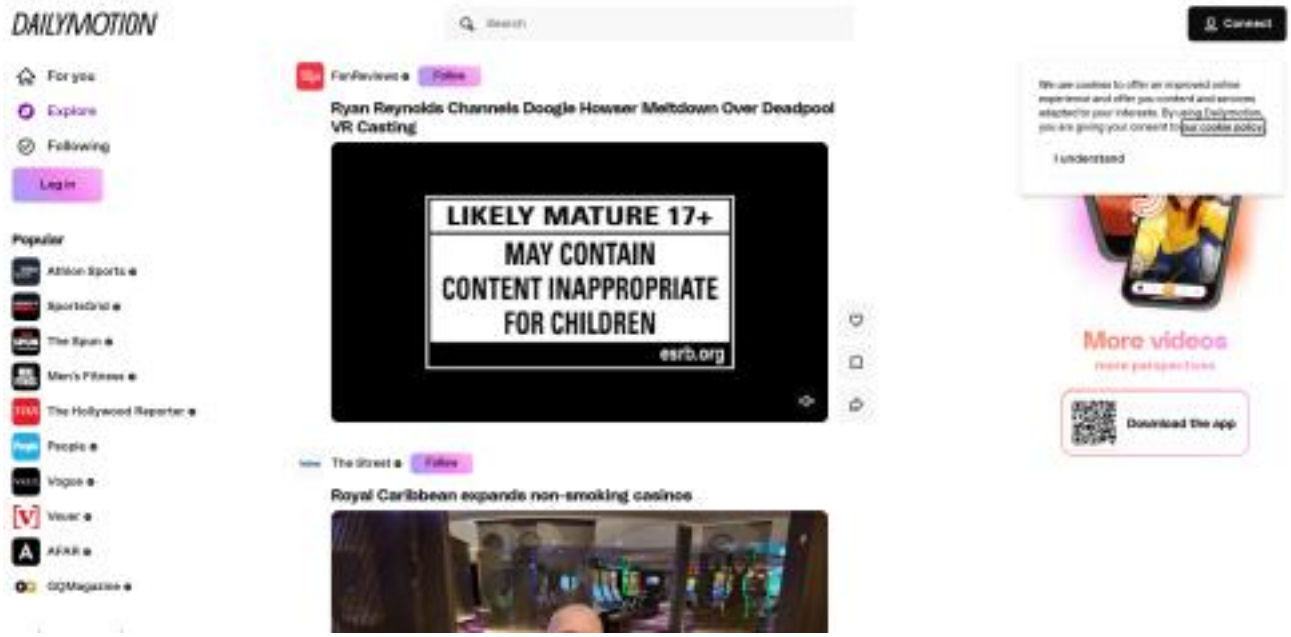


Рисунок 1.3 – головна сторінка *DailyMotion*

JW Player — це програмне рішення, яке дає змогу сайтам самостійно розміщувати та відтворювати відео напряму на своїх сторінках, без необхідності вивантаження контенту на сторонні платформи. На відміну від популярних методів вбудовування відео з таких сайтів, як *YouTube* чи *Vimeo,* цей плеєр дозволяє зберігати повний контроль над відеофайлами, що значно спрощує управління монетизацією та забезпечує більш стабільну та контрольовану доставку відеоконтенту.

Сервіс підтримує сучасні формати та забезпечує високу якість відтворення навіть за умов слабкого інтернет-з’єднання. Його зручна панель керування дозволяє управляти відображенням плеєра, додавати субтитри, інтегрувати рекламу, аналітику переглядів та інші інтерактивні елементи.

Завдяки своїй продуктивності та гнучкості *JW Player* широко використовується великими інформаційними та медіаресурсами, серед яких такі імена, як *Fox, Vice, Eurosport, Univision, Fandom* тощо. Платформа особливо популярна серед компаній,

які потребують прямої доставки відео з максимальним контролем і глибокою аналітикою для моніторингу ефективності контенту.

Крім того, *JW Player* дозволяє адаптуватися до мобільних пристроїв, що робить його зручним для сучасних сайтів із адаптивним дизайном. Завдяки потужному *API* розробники можуть налаштовувати взаємодію з відео відповідно до потреб конкретного сайту чи продукту.

Окрім базових можливостей, *JW Player* вирізняється своєю високою швидкістю завантаження, оптимізацією під різні браузери та платформами потокового

передавання. На рисунку 1.4 зображено головну сторінку *JW Player*.

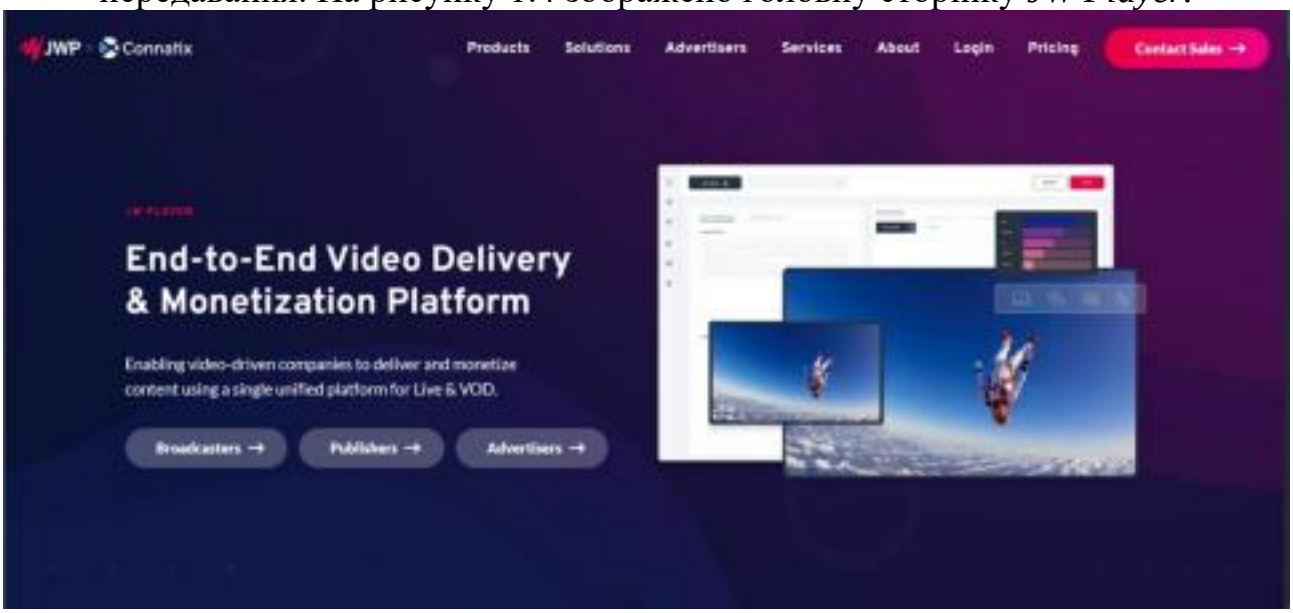


Рисунок 1.4 – Головна сторінка *JW Player*

Для *YouTube* використовується спеціальне вбудовування відео за допомогою *iframe*. Відео ідентифікується унікальним кодом, який можна знайти у посиланні на відео. Цей код підставляється у спеціально сформовану адресу, яку можна вставити в *HTML*. Також можна налаштувати параметри, наприклад автозапуск, вимкнення рекомендацій тощо.

Vimeo дозволяє імпортувати відео в подібний спосіб. Вони також генерують *iframe* з унікальним числовим ідентифікатором відео. Платформа підтримує більшу роздільну здатність, зокрема *4K*, і дозволяє керувати виглядом та взаємодією плеєра через додаткові параметри в *URL*. Додаткові налаштування можуть контролювати колір інтерфейсу, автозапуск, звук, петлю, показ аватара тощо.

Dailymotion надає свій формат для вбудовування відео у вебсторінки. Їхній плеєр також вбудовується за допомогою *iframe* з посиланням на відео, що містить унікальний ідентифікатор. Також є можливість вмикати або вимикати деякі функції, як-от автовідтворення або повноекранний режим.

Усі три сервіси дають можливість швидко інтегрувати відеоконтент у свій вебзастосунок без потреби зберігати відеофайли локально. Достатньо скопіювати відповідний фрагмент із сервісу або сформувати його вручну на основі ID відео. Якщо потрібно — можна налаштувати розміри, взаємодії, вигляд та адаптивність плеєра.

Щоб розмістити відео з *JW Player* на своєму сайті, треба завантажити відео у їхню хмарну бібліотеку, налаштувати вигляд плеєра, згенерувати спеціальний скрипт або *HTML*-код і вставити його у структуру своєї сторінки. Це дає змогу

використовувати власне відео, без сторонніх сервісів, при цьому отримуючи стабільне відтворення та аналітику.

Після завантаження відео платформа створює унікальний *ID,* а також надає код для вставки (*embed*), який можна вставити у *HTML*-код сайту. Плеєр автоматично підтягнеться і покаже обране відео. *JW Player* також дозволяє кастомізувати поведінку відтворення: автозапуск, відображення субтитрів, вибір якості відео та вивід логотипу.

Для більш досвідчених розробників існує *API* — тобто набір інструментів для взаємодії з плеєром через *JavaScript.* Це дає змогу створювати розумні плеєри, що реагують на дії користувача або взаємодіють з іншими частинами сайту.

1.2 Вимоги до функціональності інформаційного ресурсу

Вимоги до функціональності інформаційного ресурсу — це перелік основних можливостей, які повинен мати вебсайт чи застосунок, щоб ефективно виконувати своє призначення, бути зручним у користуванні та задовольняти потреби цільової аудиторії. Залежно від тематики сайту (бібліотека, блог, новинний портал, відеосервіс, навчальна платформа тощо), вимоги можуть варіюватись.

Інформаційний ресурс повинен забезпечувати повноцінну взаємодію користувача з вебзастосунком, при цьому залишаючись зручним, швидким, зрозумілим і стабільним. Однією з основних вимог є наявність системи керування контентом, яка дозволяє додавати, змінювати, редагувати, видаляти та структурувати матеріали в межах сайту. Уся інформація має бути організована логічно, розподілена за категоріями, тематиками чи рубриками, щоб користувач міг швидко знаходити потрібне. Важливо також реалізувати потужний пошук за ключовими словами, назвою, автором, тегами або датою публікації. Це дозволяє користувачеві з мінімальними зусиллями отримати доступ до необхідного контенту.

Ще однією критичною вимогою є зручна система навігації. Вебресурс повинен мати просте, інтуїтивне меню, яке залишатиметься зрозумілим незалежно від кількості сторінок. Добре продумані переходи між розділами, наявність «хлібних крихт» (*Breadcrumbs*) та кнопок швидкого повернення полегшують переміщення сайтом, що безпосередньо впливає на користувацький досвід. Для сайтів із великою кількістю матеріалів слід реалізовувати фільтрацію, сортування, пагінацію та інші

способи навігації по обширному вмісту.

Адаптивність — обов’язкова характеристика сучасного вебзастосунку. Сайт повинен коректно відображатися на різних пристроях — смартфонах, планшетах, ноутбуках і великих моніторах. Для цього використовуються адаптивні сітки, гнучкі блоки та автоматичне налаштування розміру шрифтів і зображень. Важливо, щоб інтерфейс залишався зрозумілим навіть при перегляді з екранів із малим розміром.

Ще одним компонентом є система користувачів. Реєстрація, вхід у профіль, відновлення пароля, особистий кабінет, де зберігається історія переглядів або улюблені матеріали — усе це дозволяє зробити сайт персоналізованим. Можна також реалізувати можливість коментування, обговорень, додавання оцінок або скарг. Ролі користувачів (звичайний гість, зареєстрований користувач, модератор, адміністратор) дозволяють налаштувати різні рівні доступу до інформації.

Ще однією важливою функціональною вимогою є інтеграція зі сторонніми платформами. Сучасний вебресурс повинен підтримувати імпорт відео з платформ на

кшталт *YouTube, Vimeo, Dailymotion* або *JW Player*. Це дозволяє значно розширити мультимедійні можливості сайту. Також варто реалізувати підтримку соціальних мереж — інтеграція з *Facebook, Instagram, Telegram, TikTok* дає змогу не тільки поширювати контент, а й взаємодіяти з аудиторією.

Керування безпекою — ще один критичний елемент функціональності. Захист особистих даних користувачів, реалізація безпечного з’єднання через *HTTPS,* використання *CAPTCHA*, обмеження доступу до адмінпанелі, контроль підозрілої активності — усе це необхідно для запобігання несанкціонованому доступу або злому.

Функціональність повинна включати можливість масштабування. Це означає, що ресурс повинен справлятися з великою кількістю відвідувачів без втрати швидкості роботи або стабільності. Йдеться не лише про кількість користувачів, а й про здатність розширювати функціонал — наприклад, додавати нові модулі, сторінки, інтеграції без переробки основної структури сайту.

Підтримка статистики та аналітики є також важливою функціональною вимогою. Адміністрація ресурсу має мати змогу відстежувати кількість відвідувачів, їхню поведінку, найпопулярніші сторінки, джерела трафіку, середній час перегляду. Для цього можна використовувати такі сервіси, як *Google Analytics* або вбудовані

панелі статистики.

Вебзастосунок повинен мати швидкий час завантаження, оптимізовані зображення, кешування контенту, асинхронну підгрузку блоків і можливість використання технологій *CDN* для розповсюдження контенту по всьому світу. Чим швидше працює сайт — тим нижча ймовірність втрати відвідувача. У сучасних умовах очікування більше ніж 3 секунд часто змушує користувача залишити сторінку.

Також важливим є резервне копіювання та відновлення даних. Якщо сайт містить велику кількість унікального контенту чи працює з базами даних, система повинна регулярно створювати резервні копії, аби у разі збою відновити інформацію без втрат.

Функціональність інформаційного ресурсу має враховувати не лише базові потреби користувачів, а й можливості подальшого розвитку та модернізації. Усе повинно працювати злагоджено, надійно та передбачувано.

Функціональність інформаційного ресурсу повинна бути побудована таким чином, щоб забезпечити не лише базову подачу інформації, але й максимальну інтерактивність, зручність та ефективність для кінцевого користувача. Сайт повинен слугувати не просто статичним джерелом даних, а повноцінною системою, що дозволяє обмінюватися, зберігати, переглядати, аналізувати та управляти інформацією на різних рівнях.

Однією з важливих характеристик функціонального ресурсу є можливість персоналізації. Це включає адаптацію контенту під конкретного користувача відповідно до його вподобань, минулої активності або заданих фільтрів. Наприклад, інформаційна стрічка новин може змінюватися в залежності від інтересів або попереднього пошуку користувача. Персоналізація також може передбачати збереження уподобань (темна тема, мова інтерфейсу, улюблені матеріали тощо).

Інтерактивність також має велике значення. Можливість коментування, участі в опитуваннях, голосування, оцінювання матеріалів, написання відгуків — усе це сприяє підвищенню активності аудиторії. Вебресурс повинен забезпечувати двосторонню комунікацію — не лише транслювати інформацію, а й давати змогу користувачам висловлювати думки, залишати повідомлення або взаємодіяти з адміністрацією.

Функціональний сайт повинен мати інтеграцію з базами даних. Це дозволяє

зберігати велику кількість структурованої інформації — акаунти, публікації, медіа, історію взаємодій, дані з форм, статистику тощо. База даних повинна бути добре організованою, оптимізованою, захищеною та підтримувати швидкий доступ до інформації навіть при високих навантаженнях.

Крім зберігання, функціональний ресурс повинен уміти працювати з динамічними даними. Наприклад, новини можуть завантажуватися в реальному часі без оновлення сторінки, користувачі можуть отримувати повідомлення про нові події,

зміни або коментарі. Для цього використовуються технології, як-от *AJAX, WebSocket, REST API* або *GraphQL.*

Система повідомлень — ще одна важлива функціональна складова. Вона може бути реалізована у вигляді *email*-розсилок, *push*-сповіщень у браузері, повідомлень у самому кабінеті користувача або інтеграції з месенджерами. Це дозволяє своєчасно інформувати аудиторію про нові матеріали, події, зміни або персональні оновлення.

Підтримка мультимедіа також входить до функціональних вимог. Сайт має відтворювати відео, аудіо, підтримувати галереї зображень, документи (наприклад, *PDF*), анімацію або інтерактивні карти. Це розширює подачу інформації, робить її більш візуальною та зрозумілою. Водночас мультимедійні файли мають бути оптимізованими для швидкого завантаження.

Пошукова оптимізація — ще один невід’ємний елемент функціональності. Ресурс має підтримувати структурування *URL*-адрес, метатеги, описання сторінок, заголовки, альтернативний текст для зображень — усе це допомагає пошуковим системам краще індексувати сайт і підвищує його видимість у результатах пошуку. *SEO*-функціонал має бути доступний для кожної нової сторінки або елементу контенту.

Окрему увагу слід приділяти багатомовності. Якщо сайт орієнтований на міжнародну аудиторію, він має підтримувати переклад не лише статичних текстів, але й динамічного контенту, коментарів, кнопок, системних повідомлень. На рисунку 1.5 зображено пошукову функцію знаходження інформації.

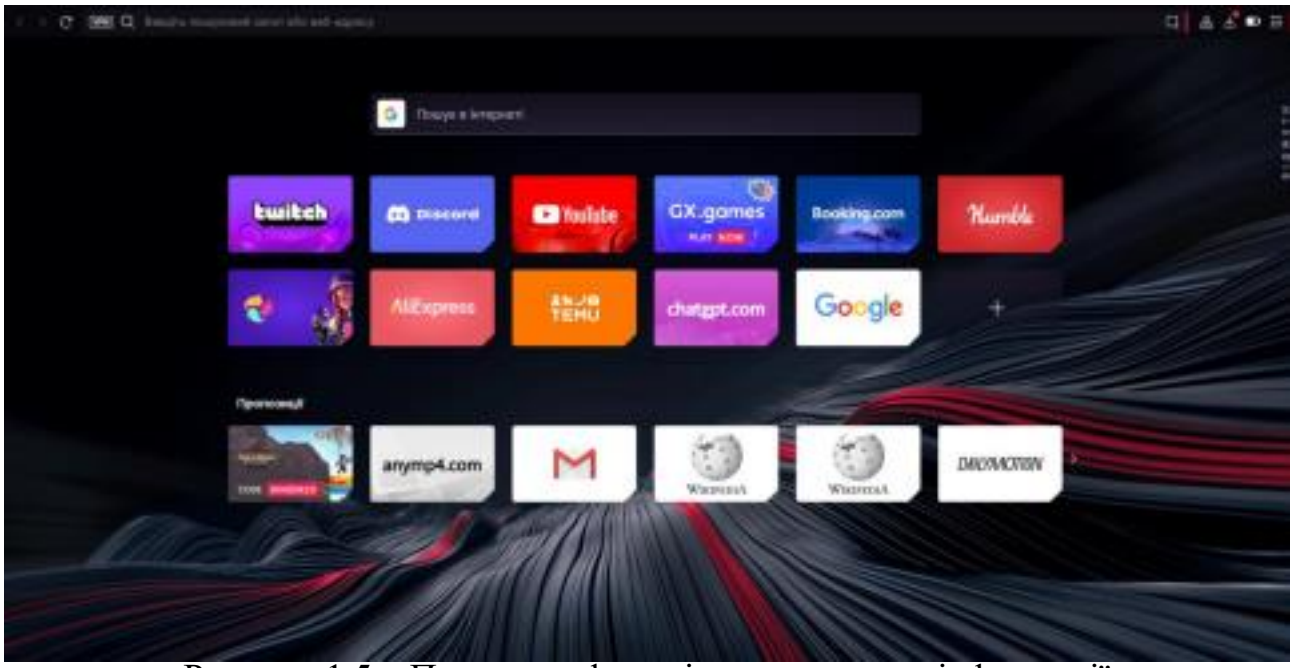


Рисунок 1.5 – Пошукова функція знаходження інформації

Функціональність інформаційного ресурсу — це не просто набір технічних можливостей, а комплексне рішення, яке забезпечує зручність, безпеку, доступність, швидкість, стабільність і перспективу розвитку. Справжній інформаційний ресурс повинен не лише інформувати, а й сприяти взаємодії, зацікавленості та довірі користувачів.

Платформа повинна забезпечувати безпеку даних користувачів, захист від несанкціонованого доступу, резервне копіювання та стабільну роботу навіть при високих навантаженнях. Інтеграція з зовнішніми системами через API дає змогу розширювати функціонал без втрати гнучкості.

Інформаційний ресурс має бути інтуїтивно зрозумілим для користувачів, що забезпечує комфортну взаємодію, навіть якщо людина вперше потрапляє на сайт. Усі ці функціональні вимоги спрямовані на те, щоб ресурс не просто зберігав або показував інформацію, а й ефективно працював із нею, був зручним, швидким і корисним.

Інформаційний ресурс повинен включати засоби збирання й обробки статистики щодо переглядів, популярності матеріалів, часу перебування на сторінці. Це дозволяє краще розуміти аудиторію та вдосконалювати платформу. Контент

повинен легко оновлюватися, бути релевантним і мати інструменти контролю за актуальністю.

Управління користувачами передбачає можливість розмежування прав доступу, створення груп, ролей, обмеження або дозволи на окремі розділи. Підтримка різноманітних типів файлів і медіа дає змогу поширювати текстову,

графічну, аудіо і відеоінформацію без втрати якості та швидкості обробки.

Платформа повинна забезпечувати безпеку даних користувачів, захист від несанкціонованого доступу, резервне копіювання та стабільну роботу навіть при високих навантаженнях. Інтеграція з зовнішніми системами через API дає змогу розширювати функціонал без втрати гнучкості.

1.3 Постановка задачі та вибір мов (*HTML,PHP,JS,CSS*)

комп’ютером. Вона дозволяє описувати послідовності дій, які має виконати машина, шляхом запису коду згідно з визначеними правилами. Кожна мова програмування має власний синтаксис — набір правил, що визначає правильну побудову інструкцій, і семантику — зміст цих інструкцій. Програмування дозволяє реалізовувати алгоритми, працювати з логічними умовами, виконувати обчислення, керувати потоками даних, обробляти події, керувати пам’яттю, будувати взаємозв’язки між компонентами системи.

Мови програмування відрізняються за рівнем абстракції. Низькорівневі мови ближчі до машинного коду і дозволяють точне управління апаратним забезпеченням. Високорівневі мови приховують технічні деталі роботи процесора і надають людині зручний інструмент для вирішення завдань. Серед можливостей мов програмування — створення змінних для збереження значень, використання умовних операторів для прийняття рішень, реалізація повторюваних дій через цикли, модульність коду через функції та методи, обробка помилок, взаємодія з користувачем, створення та зміна даних у базах даних, робота з мережами, файлами, сенсорами тощо.

Мови програмування також класифікуються за парадигмами. Існують процедурні, об’єктно-орієнтовані, функціональні, декларативні, логічні парадигми. Кожна з них пропонує певний підхід до структурування коду та організації логіки програми. Багато сучасних мов поєднують кілька парадигм. Також мови можуть бути компільованими або інтерпретованими. У компільованих мовах код спочатку перекладається у машинний код перед виконанням, а в інтерпретованих — виконується поступово, рядок за рядком, без попередньої компіляції.

Розробка програмного забезпечення вимагає точності, оскільки навіть одна помилка в синтаксисі чи логіці може призвести до неправильного виконання або зупинки програми. Саме тому навчання мові програмування вимагає уваги до

деталей, послідовного мислення, вміння розбивати завдання на підзадачі, аналізувати дані та перевіряти правильність рішень.

Перші спроби програмування з'явилися ще задовго до створення сучасних комп'ютерів. Уже в XIX столітті використовувалися пристрої, які могли виконувати заздалегідь визначені дії — наприклад, ткацькі верстати з перфокартами або піаніно, що відтворювали мелодії завдяки спеціальним механізмам. Принцип їхньої роботи нагадує сучасні предметно-орієнтовані мови програмування. На початку XX століття почали активно застосовувати перфокарти для автоматизованої обробки інформації.

HTML

Мова розмітки гіпертексту *HTML* була створена британським науковцем Тімом Бернерсом-Лі приблизно у період між 1986 і 1991 роками в науковому центрі *CERN*, що розташований у Женеві, Швейцарія. Її розробляли як інструмент для обміну технічною та науковою документацією, яким могли б користуватися навіть ті, хто не мав досвіду у візуальному оформленні текстів. *HTML* вирішував складнощі, пов'язані з використанням *SGML,* завдяки обмеженому, але функціональному набору структурних і семантичних елементів, відомих як теги.

HTML дозволяв легко створювати прості, але структуровані документи з базовим оформленням. Однією з ключових можливостей мови стало впровадження

гіпертексту — можливості переходів між документами. З часом *HTML* почав підтримувати й мультимедійний контент.

Першим відкритим описом мови став документ під назвою «Теги *HTML*», який Тім Бернерс-Лі опублікував у мережі наприкінці 1991 року. У ньому розглядалося 18 базових елементів, на основі яких було побудовано першу версію *HTML*. Більшість із цих елементів запозичено з внутрішнього формату SGML, який використовувався в *CERN*. Навіть у версії *HTML* 4, яка з'явилася пізніше, 11 з цих елементів залишаються актуальними.

Первісна концепція *HTML* передбачала створення універсального засобу для форматування текстів, що могли б однаково добре відображатися на різних пристроях — від настільних комп'ютерів до мобільних телефонів чи голосових систем. Проте з плином часу використання мови змінилося: деякі елементи, як-от , спершу призначені для побудови таблиць, почали використовуватися для візуального розміщення об'єктів на сторінці. Унаслідок цього універсальність і платформна незалежність *HTML* частково поступилися місцем новим вимогам до

дизайну, інтерактивності й мультимедіа.

На рисунку 1.6 зображено структуру *HTML* документа .

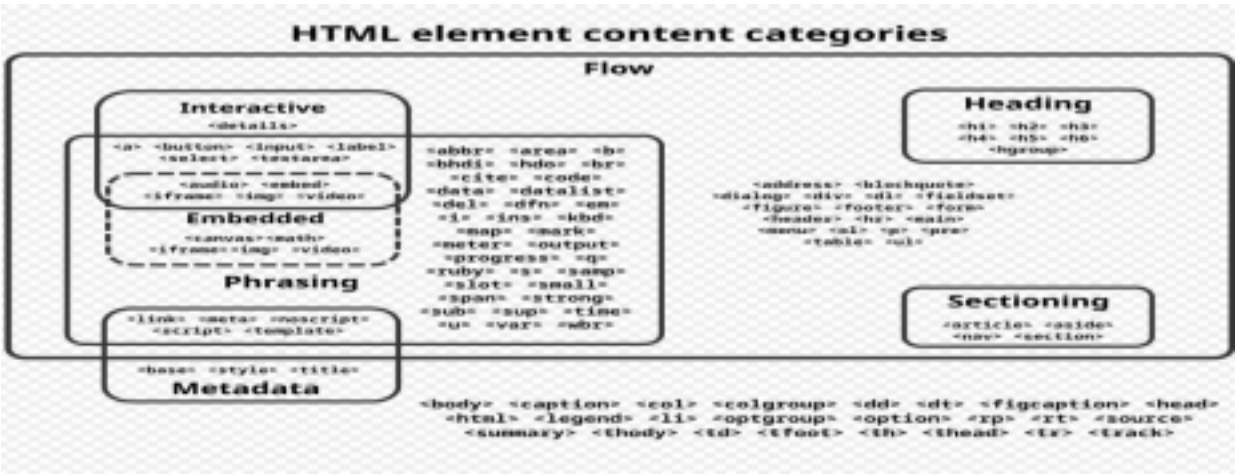


Рисунок 1.6 – Структура *HTML* документа

PHP

PHP, або *Hypertext Preprocessor* (гіпертекстовий препроцесор), раніше відомий як *Personal Home Page Tools*, — це популярна скриптова мова програмування, яка з самого початку створювалась для формування *HTML*-сторінок безпосередньо на

стороні сервера. На сьогодні *PHP* належить до числа найуживаніших мов у веброзробці, нарівні з такими як *Java, .NET, Python, JavaScript* і *Ruby*. Цю мову підтримує більшість хостинг-платформ, і вона є частиною відкритого програмного забезпечення, доступного безкоштовно для всіх охочих.

На відміну від *JavaScript, PHP* не виконується у браузері, а обробляється сервером: результатом роботи є звичайний *HTML*-код, який надсилається користувачеві. Завдяки цьому *PHP*-код залишається прихованим, що підвищує рівень безпеки. Водночас *PHP* можна використовувати для створення *JavaScript*-коду, який уже працює на клієнтському боці, що забезпечує гнучкість у побудові вебінтерфейсів.

Історія *PHP* бере початок у 1995 році, коли канадський розробник Расмус Лердорф створив невелику програму мовою *Perl* для відстеження відвідувань сторінки його резюме. З часом усе більше людей зацікавилися цим інструментом, тому він виклав його для публічного використання під назвою *Personal Home Page Tools* (Інструменти для персональної домашньої сторінки). Це стало відправною точкою розвитку *PHP*.

Невдовзі потреба в розширенні можливостей програми змусила Лердорфа переписати її на мові C. Так з'явилася друга версія — *PHP/FI* (*Forms Interpreter*, або Інтерпретатор Форм), яка вже мала базовий синтаксис, схожий на сьогоднішній *PHP*.

Ця версія дозволяла вбудовувати *PHP*-код безпосередньо в *HTML*, працювала з формами, взаємодіяла з базами даних і могла бути інтегрована з *Apache*, забезпечуючи високу продуктивність. До 1997 року *PHP* уже застосовувався на понад 50 тисячах сайтів, що хоч і складало невелику частку інтернету, проте демонструвало стрімке зростання популярності мови.

Ефективність є ключовим аспектом у програмуванні, особливо для середовищ із великою кількістю користувачів, до яких належить і веброзробка. Однією з суттєвих переваг мови *PHP* є те, що вона інтерпретована. Завдяки цьому *PHP* сценарії можуть виконуватись із доволі високою швидкістю. Згідно з окремими дослідженнями, більшість невеликих *PHP*-програм працюють швидше, ніж подібні рішення, створені на *Perl*.

Однак, незважаючи на оптимізацію, яку впроваджують розробники *PHP*, компільовані програми завжди залишатимуться продуктивнішими: їхнє виконання може бути у десятки або навіть сотні разів швидшим.

JS(JavaScript)

JavaScript (скорочено *JS*) — це багатопарадигмова мова програмування, що підтримує об'єктно-орієнтований, імперативний і функціональний підходи. Вона є реалізацією стандарту *ECMAScript (ECMA-262)* і найчастіше використовується у веброзробці для створення інтерактивних елементів на сторінках у браузері.

JavaScript надає динамічну та слабку типізацію, автоматичне управління пам'яттю, прототипне програмування, а функції в цій мові є об'єктами першого класу. Мова розроблялася з урахуванням впливу Java та інших мов програмування, щоб бути водночас знайомою й доступною для розробників. Її особливістю є те, що вона не належить жодній окремій компанії, на відміну від багатьох інших мов, поширених у вебсередовищі.

Історія *JavaScript* бере початок з середини 1990-х років. Ще в 1992 році компанія *Nombas* працювала над створенням мови *Cmm* (або *C-minus-minus*), що мала замінити макроси, залишаючись схожою на C, але з автоматичним управлінням пам'яттю. Пізніше *Cmm* була перейменована в *ScriptEase,* а на її основі створено продукт *CEnvi*, який дозволяв додавати скрипти у вебсторінки. Приклади таких сторінок, відомих як *Espresso Pages,* демонстрували можливості скриптів у браузері, хоч і працювали лише у 16-бітному *Netscape Navigator*.

Перша офіційна реалізація *JavaScript* була створена Бренданом Ейхом у

компанії *Netscape*. Цей варіант отримав назву *SpiderMonkey* і був написаний мовою C/C++. Згодом з'явився і *JavaScript*-рушій *Rhino* на основі *Java*, створений Норрісом Бойдом. Обидві реалізації підтримують стандарт *ECMAScript* і продовжують розвиватися.

JavaScript став ключовим інструментом у веброзробці завдяки своїй здатності динамічно змінювати вміст сторінки без необхідності її повного перезавантаження. Це зробило можливим створення сучасних вебзастосунків — зручних, інтерактивних

і швидких. Його гнучкість дозволила створювати як прості скрипти перевірки форм, так і складні інтерфейси користувача для великих систем.

Сьогодні *JavaScript* використовується не лише в браузерах, але й на стороні сервера завдяки середовищу *Node.js*. Таким чином, мова стала універсальним рішенням для повного циклу розробки (*frontend* + *backend*). Вона також підтримується величезною екосистемою фреймворків та бібліотек (*React, Angular, Vue, Express* тощо), що значно розширює її можливості.

JavaScript постійно еволюціонує, отримуючи нові функції, що робить його ще потужнішим і привабливішим для розробників. Його відкритість і активна *CSS*

CSS (каскадні таблиці стилів) — це мова стилізації, яка відповідає за зовнішній вигляд вебсторінок, створених мовами розмітки, такими як *HTML* чи *XML*. Вона є однією з трьох основних технологій Всесвітньої павутини нарівні з *HTML* і *JavaScript.*

Основне призначення *CSS* — відокремити структуру документа від його візуального оформлення. Завдяки цьому дизайн можна змінювати незалежно від вмісту сторінки, що робить розробку значно зручнішою та ефективнішою. *CSS* дозволяє задавати кольори, шрифти, розміщення елементів, а також адаптувати вигляд сторінки до різних типів пристроїв — від комп'ютерів і смартфонів до принтерів та екранів Брайля.

РОЗДІЛ 2
ПРОЕКТУВАННЯ СИСТЕМИ ВЕБ-ЗАСТОСУНКУ

2.1 Архітектура веб-застосунку

Архітектура веб-застосунку — це структура, яка визначає, як компоненти

вебсайту або вебдодатку взаємодіють між собою, як організовано обробку запитів користувача, зберігання й обробку даних, а також передачу контенту на клієнтський пристрій. Вона включає як клієнтську, так і серверну частину, які працюють разом для забезпечення функціональності вебресурсу. Основна мета архітектури — забезпечити ефективність, масштабованість, безпеку й зручність у підтримці застосунку. У типовому вебдодатку є фронтенд (що бачить користувач: *HTML, CSS, JavaScript*), бекенд (серверна логіка: *PHP, Node.js, Python* та інше), база даних (*MySQL, MongoDB, JSON*-файли тощо), а також мережеві запити між цими компонентами.

У вебзастосунку реалізовано стандартну клієнт-серверну архітектуру. Клієнтська частина представлена набором *HTML*-сторінок, стилів *CSS,* зображень та JavaScript-коду. Вона відповідає за візуальну частину інтерфейсу, відображення контенту та взаємодію з користувачем у браузері. Усі *HTML*-файли є окремими самостійними сторінками, кожна з яких має своє призначення. Наприклад, є окремі сторінки для входу, реєстрації, головної сторінки, а також по одній сторінці для кожного мультимедійного елемента, що демонструє трейлер.

Зовнішній вигляд сайту оформлюється за допомогою одного *CSS*-файлу. Він містить стилі для розмітки, розташування елементів, фонових зображень, шрифтів, кольорової схеми та інших параметрів презентації. Всі сторінки підключають цей файл для уніфікації дизайну.

JavaScript-файл використовується для додавання інтерактивності. Наприклад, він може відповідати за обробку кліків, перемикання слайдів або взаємодію з

формами. У структурі є також набір зображень: фонова графіка, зображення для слайдера та прев'ю до кожного окремого мультимедійного елементу. Серверна частина побудована на *PHP*. Вона виконує обробку даних із форм, таких як реєстрація та вхід користувачів. Для цього створено відповідні скрипти, які обробляють *POST*-запити від клієнта, перевіряють введену інформацію, а також читають і змінюють дані у файлі формату *JSON. JSON*-файл виконує функцію локального сховища — у ньому зберігається масив зареєстрованих користувачів, їх логіни, паролі та інша інформація.

PHP-файл, який відповідає за реєстрацію, отримує дані з форми, перевіряє їх на коректність і, у разі успіху, додає новий запис до *JSON*-файлу. Аналогічно, скрипт входу читає цей файл, порівнює надані облікові дані з наявними і виконує

відповідні дії — наприклад, допускає до внутрішніх сторінок або виводить повідомлення про помилку.

Також присутній окремий скрипт, який може надавати доступ до всього списку користувачів або певної інформації з *JSON*-файлу. Усі ці файли знаходяться в одній директорії, яка є коренем для застосунку. Таке розміщення забезпечує просту й зрозумілу структуру: всі ресурси згруповані за типами, доступ до них здійснюється напряму.

Уся архітектура розрахована на те, щоб клієнтська частина взаємодіяла із серверною через звичайні *HTTP*-запити без використання сторонніх фреймворків чи баз даних. Це дозволяє розгорнути застосунок у будь-якому середовищі, що підтримує *PHP*, наприклад, у локальному сервері типу *XAMPP*.

2.2 Створення бази даних

База даних — це впорядковане сховище інформації, яке дозволяє ефективно зберігати, шукати, оновлювати та видаляти дані. Вона служить основою для зберігання великого обсягу структурованої або напівструктурованої інформації, до якої можна звертатися за допомогою спеціальних мов запитів або програмного коду.

У вебзастосунках база даних зазвичай використовується для збереження облікових даних користувачів, інформації про товари, замовлення, публікації, повідомлення тощо.

Бази даних поділяються на кілька основних типів, залежно від способу організації та зберігання інформації:

Кожен запис має унікальний ідентифікатор — первинний ключ. Таблиці можуть бути пов’язані між собою за допомогою зовнішніх ключів. Реляційні бази підтримують мову *SQL* (*Structured Query Language*), Приклади: *MySQL, PostgreSQL, SQLite, Microsoft SQL Server, Oracle Database*.

Документно-орієнтовані бази даних (*Document-oriented Databases*) Цей тип баз використовує зберігання даних у вигляді документів, зазвичай у форматах *JSON, BSON* або *XML*. Документи можуть мати гнучку структуру, що дає змогу зберігати різні поля для різних записів без жорсткої схеми, як у реляційних базах.

Підходить для зберігання складних або вкладених структур даних. Приклади: *MongoDB, CouchDB, Amazon DocumentDB*.

Ключ-значення(*Key-Value-Stores*)

Ці бази зберігають інформацію у вигляді пари ключ — значення. Ключ виступає унікальним ідентифікатором, а значення може бути будь-яким об’єктом або рядком. Вони надзвичайно швидкі й прості, підходять для кешування або зберігання сесій. Приклади: *Redis, Amazon DynamoDB, Riak.*

Колонкові бази даних *(Column-oriented Databases)* На відміну від реляційних баз, де дані зберігаються по рядках, колонкові бази зберігають інформацію по стовпцях. Це дозволяє ефективно обробляти великі обсяги аналітичних запитів. Переважно використовуються у системах аналітики. На рисунку 2.1 зображено робочу студію *My SQL* .

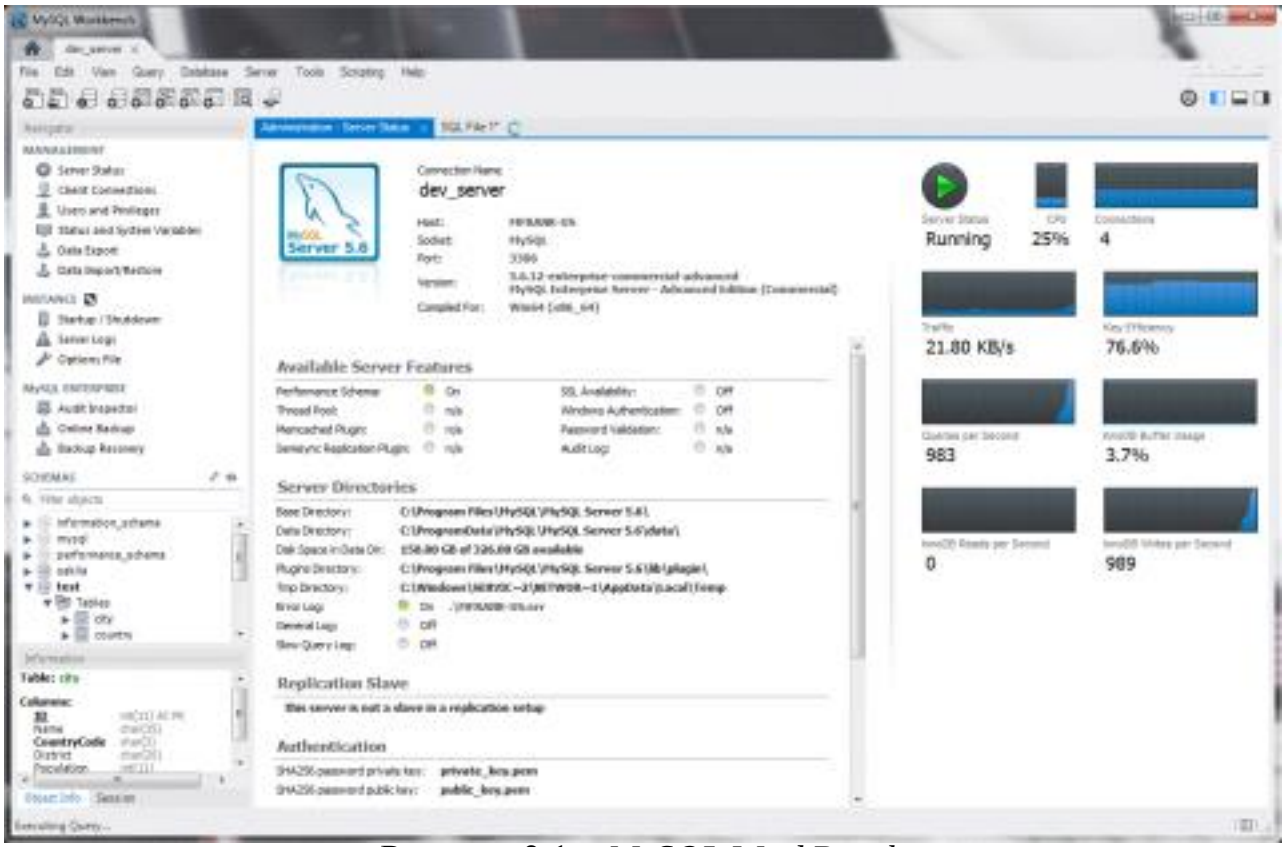


Рисунок 2.1 – *MySQL WorkBench*

База даних, реалізована у цьому вебзастосунку, є не традиційною реляційною базою, як-от *MySQL* або *PostgreSQL*, а файлом у форматі JSON, який виступає у ролі основного сховища всіх користувацьких облікових даних. Цей підхід належить до категорії файлових систем зберігання даних і широко використовується у навчальних проєктах, тестових середовищах або у невеликих вебзастосунках, де масштабованість, складна структура запитів або великі обсяги даних не є критично важливими. Файл users.json є ключовим компонентом — у ньому зберігаються всі записи зареєстрованих користувачів у вигляді серіалізованих об'єктів.

У процесі реєстрації, після того як користувач заповнює відповідну форму на

сайті, серверний *PHP*-скрипт отримує значення, передані через метод *POST*. Далі скрипт виконує відкриття файлу users.json для читання. Весь вміст файлу завантажується у вигляді тексту, після чого декодується у внутрішній формат *PHP* — масив об’єктів, кожен з яких містить інформацію про одного користувача: ім’я, електронну пошту та пароль. Ці дані далі обробляються — додається новий

користувач як новий елемент до масиву, після чого весь масив знову серіалізується у *JSON* і перезаписується у файл.

Уся логіка роботи побудована навколо цього файлу. Наприклад, для входу користувача (автентифікації) знову відкривається файл, зчитується весь його вміст, декодується, і *PHP* перебирає всі збережені записи у пошуках збігу логіна та пароля. Важливо підкреслити, що у цій системі паролі зберігаються у відкритому вигляді, тобто без хешування, що є критичним з погляду безпеки у реальних комерційних проєктах, але у навчальному контексті це дозволяє краще зрозуміти принципи обробки даних. У разі виявлення збігу користувач отримує доступ до сайту.

Файл бази даних *users.json* функціонує як централізоване місце, де зберігається повна історія користувацької активності у плані реєстрацій. Усі дії, пов’язані із створенням облікових записів, перевіркою користувача, потенційно — зміною пароля або видаленням облікового запису — вимагають повного перечитування та перезапису цього файлу. Це означає, що навіть одна операція, наприклад оновлення електронної адреси користувача, технічно вимагає читання всього масиву, зміни одного з його елементів і запису всього *JSON*-файлу наново.

Така модель має низку особливостей. По-перше, вона не забезпечує конкурентний доступ — тобто, якщо кілька користувачів одночасно будуть вносити зміни, це може призвести до втрати частини даних через конфлікти перезапису. По друге, файл зростає зі збільшенням кількості користувачів, що потенційно може уповільнити час відкриття, зчитування та обробки. По-третє, відсутній захист від прямого доступу до файлу, якщо не реалізовано належних обмежень у конфігурації вебсерверу.

У контексті архітектури вебсайту, цей файл є частиною серверної логіки. Він не взаємодіє безпосередньо з клієнтом (браузером), але весь процес реєстрації й входу, який відбувається на стороні сервера, залежить від його вмісту. На відміну від *SQL* баз, де для кожної операції відбувається точковий запит, тут будь-яка зміна чи пошук реалізується через повне прочитання та обробку всього масиву. Це забезпечує

простоту коду, легкість розуміння структури та можливість гнучко змінювати формат збереження.

JSON як формат має свої переваги — це легкий, зручний для читання і сумісний з *JavaScript* (навіть у разі потреби клієнтської обробки). Він добре підходить для зберігання структурованих даних — з вкладеними об’єктами, масивами, текстовими та числовими значеннями. Це дозволяє при потребі розширити об’єкти користувачів новими атрибутами, такими як фото, адреса, уподобання, стан профілю тощо, без потреби повністю змінювати структуру системи.

Фактично, ця база даних у *JSON*-файлі слугує у ролі найпростішого, але самодостатнього засобу ідентифікації користувачів та їхнього контролю доступу. Вона пов’язана безпосередньо з формами на сайті, логікою PHP-обробки запитів та контролем сесій. Відсутність складної схеми з таблицями, зовнішніми ключами або обмеженнями робить систему гнучкою й максимально адаптованою для швидкої розробки й експериментів.

У перспективі така база може бути мігрувана до повноцінної СУБД — при цьому її логіка може бути збережена, а структура файлу — використана як основа для побудови таблиць та запитів. Але на нинішньому етапі вона повністю виконує роль ядра облікової системи, забезпечуючи можливість реєстрації, входу та зберігання даних у простому й наочному вигляді.

Система управління базами даних та *MySQL*

Система управління базами даних є програмним забезпеченням яке дозволяє працювати з базами даних Вона відповідає за збереження структурування доступ зміну й видалення даних у зручний спосіб без необхідності взаємодіяти безпосередньо з файлами на диску СУБД реалізує інтерфейси завдяки яким програми й користувачі можуть працювати з даними через зрозумілі запити та команди

Серед найбільш поширених СУБД виділяють реляційні документоорієнтовані графові обєктноорієнтовані та інші Реляційні є найбільш відомими і зберігають дані у вигляді таблиць Документоорієнтовані працюють з документами наприклад у

форматі *JSON* або *XML* Графові бази даних зберігають обєкти і звязки між ними у вигляді графів

MySQL є прикладом реляційної системи управління базами даних Вона працює на основі мови SQL яка використовується для створення структури бази додавання редагування пошуку та видалення даних *MySQL* є відкритою безплатною для більшості випадків та має велику популярність у сфері веброзробки

Ця СУБД підтримує транзакції індекси зовнішні ключі збережені процедури тригери та представлення що дозволяє створювати складні інформаційні системи для сайтів блогів інтернетмагазинів форумів та багатьох інших типів вебзастосунків

MySQL підтримує різні типи таблиць зокрема *InnoDB* який є найпоширенішим та забезпечує надійність і цілісність даних а також дозволяє використовувати зовнішні ключі для звязку між таблицями *MyISAM* є старішим типом який є швидким проте не підтримує транзакції Інші типи таблиць включають *MEMORY* який зберігає дані в оперативній пам'яті та використовується для тимчасових обчислень *ARCHIVE* який підходить для зберігання великих обсягів даних що не змінюються *CSV* для зберігання у вигляді файлів з роздільниками та інші

Завдяки своїй стабільності підтримці великої кількості інструментів зручному адмініструванню через інтерфейси типу *phpMyAdmin* або командний рядок та широкій документації *MySQL* широко застосовується як основа інформаційної частини більшості сучасних сайтів та вебзастосунків

2.3 Структура веб-інтерфейсу

Вебінтерфейс — це складний багаторівневий механізм, що забезпечує безпосередню взаємодію користувача з вебзастосунком чи вебсайтом через браузер. Його головна функція — зробити всі можливості сайту доступними через візуальне середовище, зрозуміле для людини, незалежно від рівня її технічної підготовки. У контексті сучасного інтернету вебінтерфейс є не лише оболонкою або візуальним

представленням інформації — це повноцінний засіб управління інформаційними потоками, навігації, збору даних і їхньої візуалізації в інтерактивній формі. Сьогодні вебінтерфейси будуються на основі комбінації мов *HTML, CSS* і *JavaScript*, які разом створюють фундамент: структура, стилі й логіка. *HTML* задає каркас сторінки: це текст, форми, кнопки, поля введення. *CSS* відповідає за візуальне оформлення, задає шрифти, кольори, розміри, розміщення елементів на сторінці, а також ефекти. *JavaScript* керує динамікою та інтерактивністю: дозволяє реагувати на дії

користувача, змінювати вміст сторінки без перезавантаження, реалізовувати валідацію форм, обробку подій, а також взаємодію з сервером у режимі реального часу.

До вебінтерфейсу часто додаються технології асинхронної взаємодії із сервером — *AJAX*, що дозволяє зменшити навантаження на сервер і прискорити взаємодію. Завдяки цьому інтерфейс може оновлювати лише частину сторінки, а не перезавантажувати її цілком, забезпечуючи безперервну роботу користувача з мінімальними затримками.

Сучасні вебінтерфейси підтримують адаптивність — здатність автоматично підлаштовуватися під розмір екрану користувача: телефон, планшет, ноутбук, телевізор тощо. Для цього розробники використовують медіазапити в *CSS*, сіткові системи, гнучкі контейнери (*flexbox*), а також мобільні фреймворки, які полегшують створення адаптивного дизайну. Водночас важливо забезпечити доступність для всіх користувачів — включно з людьми з інвалідністю, для чого використовуються спеціальні атрибути (наприклад, *aria-label, role*, фокус-навігація), що дозволяють працювати з інтерфейсом за допомогою клавіатури чи екранних читалок.

З технічного боку вебінтерфейс — це лише одна частина повного технологічного стеку, який охоплює клієнтську частину (*frontend*) і серверну *(backend).* У випадку твого сайту клієнтська частина створена за допомогою *HTML, CSS* і *JavaScript,* і відповідає за всю взаємодію користувача з вебзастосунком: форми входу та реєстрації, кнопки навігації, вивід даних, пошукове поле, панелі тощо. Вебінтерфейс надсилає запити на сервер, де вони обробляються через серверні

скрипти — у твоєму випадку це *PHP*. Сервер перевіряє запити, взаємодіє з базою даних, і повертає відповідь, яку зчитує *JavaScript,* оновлюючи інтерфейс без оновлення сторінки.

На практиці вебінтерфейс твого сайту забезпечує кілька важливих функцій:
інтуїтивну навігацію сайтом, з логічною структурою і зрозумілими кнопками
збирання даних від користувача (вхід, реєстрація, інші інтерактивні дії)
відображення вмісту з бази даних у форматі, зручному для перегляду обробку взаємодії користувача з елементами інтерфейсу (натискання, введення, прокручування)

адаптивність до різних екранів (телефони, планшети, ПК)

Інтерфейс об’єднаний із серверною частиною сайту через запити, які зазвичай

виконуються методами *GET* або *POST*. Наприклад, під час реєстрації користувач вводить дані в *HTML*-форму, яка передає їх на сервер через *PHP*-обробник. Сервер виконує перевірки, записує дані в базу *(MySQL),* а потім повертає відповідь, яка або повідомляє про успішну реєстрацію, або виводить помилки у тому самому інтерфейсі.

Усе це забезпечує безперервну й плавну взаємодію без зайвих завантажень чи переходів між сторінками.

Також у твоєму вебінтерфейсі реалізовано взаємодію з обліковими записами користувачів. Завдяки цьому користувачі мають можливість входити в систему, реєструватися, а в майбутньому — керувати своїми діями на сайті. Це передбачає додаткову логіку у вебінтерфейсі, зокрема перевірку стану сесії, динамічну зміну доступних елементів інтерфейсу залежно від авторизації тощо.

Загалом вебінтерфейс є ядром взаємодії між людиною і цифровою системою сайту. Від його якості, зрозумілості, швидкості та стабільності залежить перше враження від сайту, рівень зручності користування і, в кінцевому результаті, загальна ефективність вебзастосунку.

2.4 Вибори бібліотек та сервера

Під час створення вебсайту було використано комплекс інструментів, бібліотек і технологій, які забезпечили як зовнішній вигляд інтерфейсу, так і стабільну роботу внутрішніх функціональних механізмів. Кожен компонент системи виконує свою специфічну роль і бере участь у створенні повноцінного вебзастосунку, який працює як на клієнтській, так і на серверній стороні. Для цього було задіяно спеціалізоване програмне забезпечення, яке дозволяє змоделювати серверне середовище на локальному комп’ютері, забезпечити зберігання даних, обробку запитів і зворотну передачу інформації користувачу.

Основою локального серверного середовища стала програма *XAMPP*. Це безкоштовний дистрибутив, який включає в себе *Apache* — найпопулярніший вебсервер, що відповідає за обробку *HTTP*-запитів, *MySQL* — потужну реляційну систему керування базами даних, *PHP* — серверну мову програмування для обробки логіки сайту, та *phpMyAdmin* — зручний вебінтерфейс для адміністрування бази даних. Завдяки *XAMPP* розробник може розгорнути та тестувати вебпроєкт локально, без необхідності розміщення на віддаленому хостингу, що значно

пришвидшує розробку та усунення помилок.

Інтерфейс вебсайту створено за допомогою *HTML, CSS* та *JavaScript. HTML (HyperText Markup Language)* є основною розміткою, яка формує структуру сторінки — блоки, заголовки, параграфи, форми, поля для введення, кнопки тощо. *CSS (Cascading Style Sheets)* відповідає за візуальне оформлення: кольори, шрифти, відступи, вирівнювання, розташування елементів на сторінці, анімації. *JavaScript* додає динамічності — реалізує взаємодію з користувачем, обробляє події, оновлює вміст сторінки без її перезавантаження. Разом ці технології формують повноцінний вебінтерфейс, що відповідає сучасним стандартам.

Крім базових технологій, у проєкті могли бути використані популярні бібліотеки. Наприклад, *jQuery* — це бібліотека для *JavaScript,* яка спрощує взаємодію з елементами *DOM,* дозволяє ефективно працювати з подіями, ефектами, *AJAX*-

запитами. Ще одна важлива технологія — *Bootstrap* — *CSS*-фреймворк, що дає змогу швидко створювати адаптивний дизайн завдяки готовим класам, грідам, стилізованим формам і компонентам інтерфейсу. Використання цих бібліотек дозволяє значно скоротити час на верстку та забезпечити кросбраузерну сумісність.

На серверній стороні працює *PHP*, мова сценаріїв, що виконується безпосередньо на сервері. *PHP* обробляє дані, які надсилає користувач із форм (наприклад, реєстрація, вхід), проводить валідацію, виконує запити до *MySQL*-бази даних, зберігає або отримує потрібну інформацію. База даних *MySQL* служить сховищем, де зберігається уся ключова інформація — облікові записи користувачів, паролі, електронні адреси тощо. Для зручного адміністрування структури таблиць, створення полів і перегляду записів використовувався вебінтерфейс *phpMyAdmin*. Взаємодія між цими компонентами дозволяє забезпечити повний цикл обробки запиту: від введення даних у формі до збереження і відображення відповідей на сайті. Така архітектура гарантує функціональність, масштабованість і стабільність роботи вебзастосунку навіть у локальному середовищі.

Для данного веб-застосунку використовувалася така програма як *XAMPP* На рисунку 2.2 зображено контрольну панель програми *XAMPP*.

Рисунок 2.2 – *XAMPP* контрольна панель

XAMPP — це спеціальна програма, яка створює на твоєму комп'ютері умови, схожі на справжній вебсервер. Завдяки цьому ти можеш розробляти вебсайти, робити

вебдодатки і тестувати їх, не підключаючись до Інтернету. Це дуже важливо, тому що ти можеш спокійно перевіряти і виправляти свій код, експериментувати, не боячись, що твій сайт в Інтернеті вийде з ладу або буде доступний для інших до того, як він готовий. Такий локальний сервер особливо потрібен програмістам, вебдизайнерам, студентам і всім, хто вчиться робити сайти або хоче перевірити роботу складних систем.

XAMPP — це набір кількох програм, які працюють разом. Найголовніше тут — це вебсервер, який відповідає за те, щоб приймати запити від браузера (коли ти вводиш адресу сайту) і віддавати йому сторінки. Вебсервер у *XAMPP* називається *Apache*, і це дуже поширене і надійне програмне забезпечення, яке використовується і в реальних серверах в Інтернеті.

Другий важливий компонент — це база даних. Вебсайти часто потребують зберігати багато інформації: дані про користувачів, товари, замовлення, повідомлення і так далі. Для цього потрібна система керування базами даних, яка зберігає всю цю інформацію організовано і дозволяє швидко її знаходити. В *XAMPP* за це відповідає *MySQL* або *MariaDB*. Це складна програма, яка управляє даними, і ти можеш з нею взаємодіяти через спеціальні інструменти.

Третій компонент — це *PHP*. Це мова програмування, яка виконується саме на сервері. За допомогою *PHP* сайт може змінюватися в залежності від користувача, відправляти і отримувати дані, взаємодіяти з базою даних, показувати різну

інформацію на одній і тій же сторінці і багато іншого. Саме завдяки *PHP* сайти стають динамічними, а не просто набором статичних сторінок.

Після встановлення *XAMPP* ти запускаєш цю програму, і в її панелі керування можеш вмикати або вимикати сервер і базу даних. Після запуску *Apache* серверу комп'ютер починає слухати запити від браузера на певному адресі — зазвичай це "*localhost*" або "127.0.0.1". Це спеціальна адреса, яка вказує на твій власний комп'ютер. Тобто ти можеш відкрити браузер і ввести цю адресу, щоб побачити свої сайти, які зберігаються на комп'ютері.

Файли сайту треба покласти у спеціальну папку, яка називається "htdocs" — це основна папка, з якої вебсервер бере сторінки для показу. Ти можеш створювати там свої папки для різних проектів, і тоді заходити у браузері за адресою, яка відповідає цій папці. Наприклад, якщо ти створив папку "myproject", тоді в браузері адреса буде виглядати як "localhost/myproject", і сервер покаже тобі файли саме з цієї папки.

Під час роботи з *XAMPP* дуже важливо пам'ятати, що це середовище розробки, а не справжній публічний сервер. Це означає, що твої сайти доступні лише на твоєму комп'ютері, якщо ти спеціально не налаштовуєш його і не даєш доступ іншим. Також тут не варто очікувати високої безпеки, тому що *XAMPP* налаштований максимально просто для зручності, а не для захисту від хакерів.

Якщо на твоєму комп'ютері вже є інші програми, які використовують той самий порт, що і *Apache* (порт 80 або 443), *XAMPP* може не запустити вебсервер. Це часта проблема, і її можна вирішити, змінивши порт, на якому працює сервер. Після цього в браузері доведеться вказувати новий порт у адресі, щоб побачити сайт.

Крім вебсерверу і бази даних, у *XAMPP* є інші допоміжні сервіси, які ти можеш не використовувати, але вони є — наприклад, поштовий сервер або *FTP*. Вони корисні для більш складних проектів.

Для роботи з базою даних ти можеш скористатися спеціальним графічним інтерфейсом, який вже входить в *XAMPP*. Він дозволяє легко створювати нові бази, таблиці, переглядати і редагувати дані, виконувати запити без необхідності писати все вручну. Це дуже допомагає, якщо ти ще тільки вчишся.

У процесі роботи з *XAMPP* часто доводиться налаштовувати конфігураційні файли, щоб змінювати різні параметри: наприклад, збільшувати максимальний розмір файлів, які можна завантажити на сервер, або налаштовувати час очікування відповіді від сервера, або керувати тим, як сервер працює з файлами і дозволами. Ці

налаштування дозволяють краще підлаштувати сервер під твої завдання.

РОЗДІЛ 3
СТВОРЕННЯ ВЕБ-ЗАСТОСУНКУ

3.1 Розробка *HTML* структури

HTML — це мова, яка задає структуру веб-сторінки і дозволяє браузеру зрозуміти, що і де розміщувати. Вона описує елементи, як текст, зображення, кнопки, посилання, а також їх взаємне розташування. *HTML* не займається оформленням чи поведінкою сторінки, але без неї сайт просто не може існувати.

На сайті використовується *HTML* 5

П’ята версія *HTML* обрана тому, що вона має багато переваг для створення сучасних сайтів. Вона підтримує нові семантичні теги, які роблять код більш зрозумілим і структурованим. Це допомагає браузерам, пошуковим системам і спеціальним пристроям краще розуміти вміст сторінки. Версія також дає змогу без проблем додавати мультимедійний контент, наприклад відео чи аудіо, без використання сторонніх плагінів.

Ця версія *HTML* сумісна з усіма сучасними браузерами і забезпечує адаптивність сайту — він буде коректно відображатися на різних екранах: від великих моніторів до мобільних телефонів. Крім того, вона підтримує нові можливості для створення інтерактивних елементів у поєднанні з *CSS* і *JavaScript.*

У структурі сайту є блоки, які задають базову організацію сторінки. Відповідно до логіки розміщення елементів, спочатку йде основний контейнер з різними складовими: заголовок, навігаційні кнопки, поле для пошуку. Далі — набір карток з трейлерами, які містять зображення та підпис. Кожна картка є посиланням, що веде на окрему сторінку.

Важливою частиною є модальні вікна — вони з’являються при взаємодії з певними кнопками і допомагають користувачу робити вибір або підтверджувати дії. Їх поява керується за допомогою класів та стилів, що контролюються через *JavaScript.*

Завдяки п’ятій версії *HTML* легко організувати логічну і зручну структуру, яка відповідає сучасним стандартам і кращим практикам веброзробки. Вона дозволяє

створити сайт, який швидко завантажується, має привабливий вигляд і зручний у використанні.

Обраний *HTML* допомагає досягти сумісності, доступності і функціональності без зайвих складнощів, що важливо для користувачів з різними пристроями і потребами. Водночас, він забезпечує основу для подальшого розширення сайту, додавання стилів і сценаріїв, які зроблять інтерфейс ще приємнішим і більш динамічним.

Звісно! Ось пояснення кожного рядка з відповідним кодом під ним для зручності.

На рисунку 3.1 зображено мета тег, що задає кодування символів *UTF*-8, щоб текст коректно відображався, зокрема українські літери.

```
<meta charset="UTF-8" />
```

Рисунок 3.1 – Кодування символів *UTF*-8

На рисунку 3.2 зображено мета тег, який задає адаптивність сторінки — ширина дорівнює ширині пристрою, масштаб 1.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0"/>
```

Рисунок 3.2 – Адаптивність сторінки

На рисунку 3.3 зображено заголовок сторінки, який відображається у вкладці браузера.

```
<title>Бібліотека Трейлерів</title>
```

Рисунок 3.3 – Заголовок сторінки

На рисунку 3.4 зображено підключення зовнішнього *CSS*-файлу зі стилями.

```
<link rel="stylesheet" href="style.css" />
```

Рисунок 3.4 – Підключення зовнішнього *CSS*-файлу зі стилями

На рисунку 3.5 зображено стилі для кнопок з класом *side-btn* — колір, фон, округлі краї, відступи, курсор тощо.

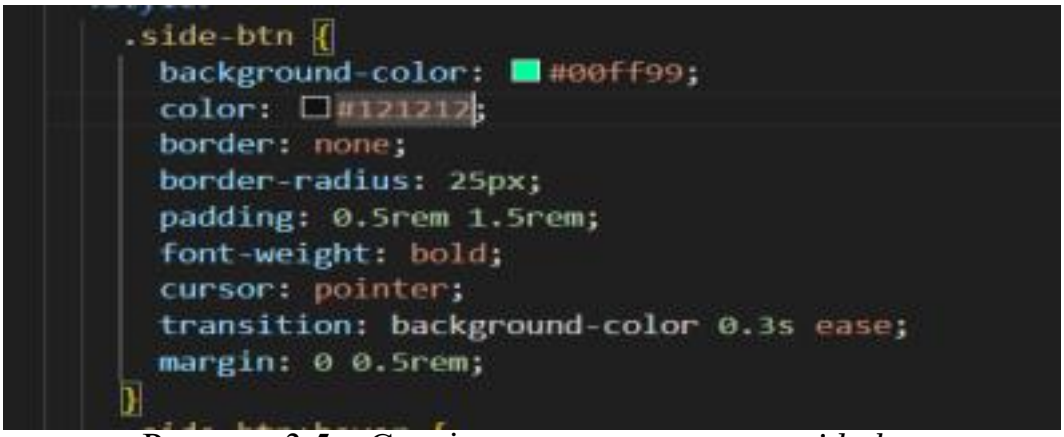


Рисунок 3.5 – Стилі для кнопок з класом *side-btn*

На рисунку 3.6 зображено стилі для кнопок *side-btn* при наведенні курсора — змінюється фон на темніший.

Рисунок 3.6 – Стилі для кнопок *side-btn* при наведенні

На рисунку 3.7 зображено стилі для модального вікна, яке спочатку не показується, а розтягується на весь екран із затемненням і розмиттям фону.

Рисунок 3.7 – Стилі для модального вікна

На рисунку 3.8 зображено модальне вікно при додаванні класу *show* — вмикає *flex*-контейнер.

Рисунок 3.8 – Показ модального вікна при додаванні класу *show*

На рисунку 3.9 зображено стилі для вмісту модального вікна — напівпрозорий фон, округлені краї, фіксована ширина з максимальною, прокрутка, тінь і кольори

тексту.

Рисунок 3.9 – Стилі для вмісту модального вікна

На рисунку 3.10 зображено стилі для кнопки закриття модалки — позиція у верхньому правому куті, великий шрифт, зелений колір, курсор при наведенні.

Рисунок 3.10 – Стилі для закриття модального вікна

На рисунку 3.11 зображено зміну контрасту при наведенні на кнопку закриття.

Рисунок 3.11 – Зміна контрасту при наведенні на кнопку закриття

На рисунку 3.12 зображено блок з трейлерами у вигляді сітки, яка гнучко розташовує картки, відступи між ними, центрований контент.

Рисунок 3.12 – Блок з трейлерами у вигляді сітки

На Рисунку 3.13 зображено картка трейлера — темний фон, округлені краї, приховує зайве, курсор при наведенні, вертикальне розміщення контенту, фіксовані розміри, плавна анімація.

Рисунок 3.13 – Стилі картки трейлера

На рисунку 3.14 зображено ефект масштабування.

Рисунок 3.14 – Збільшення картки при наведенні

На рисунку 3.15 зображено текст під зображенням — зелений, напівжирний, центрується, невеликий відступ зверху.

Рисунок 3.15 – Текст під зображенням на картці трейлера

На рисунку 3.16 зображено панель пошуку із кнопками та полем для введення тексту.

Рисунок 3.16 – Код панелі пошуку

На рисунку 3.17 зображено контейнер з посиланнями-картками на окремі сторінки трейлерів.

Рисунок 3.17 – Контейнер з посиланнями на відео

На рисунку 3.18 зображено модальне вікно для вибору цікавих трейлерів, яке приховано спочатку.

Рисунок 3.18 – Модальне вікно для вибору цікавих трейлерів

На рисунку 3.19 зображено модальне вікно для підтвердження виходу, також спочатку приховане.

Рисунок 3.19 – Модальне вікно для підтвердження виходу

На рисунку 3.20 зображено обробник події вводу в поле пошуку: отримує текст, шукає збіги у назвах карток, показує або ховає картки, показує повідомлення, якщо збігів немає.

Рисунок 3.20 – Обробник події вводу в поле пошуку

На рисунку 3.21 зображено оголошення змінних для кнопки «Цікавіше», модалки та її закриття, сітки з трейлерами.

Рисунок 3.21 – Оголошення змінних для кнопки Цікавіше

На рисунку 3.22 зображено функція відкриття модального вікна і заповнення його картками трейлерів (копіюються картки з основної сторінки).

Рисунок 3.22 – Функція відкриття модального вікна

На рисунку 3.23 зображено закриття модалки при натисканні на хрестик.

Рисунок 3.23 – Закриття модального вікна при натисканні на хрестик

На рисунку 3.24 зображено закриття модалки при кліку поза вмістом модального вікна.

Рисунок 3.24 – Закриття модального вікна по кліку

На рисунку 3.25 зображено оголошення змінних для кнопки виходу та модалки виходу, а також кнопок підтвердження/скасування.

Рисунок 3.25 – Оголошення змінних для кнопки виходу

3.2 Створення дизайну веб-сайту (*CSS*)

CSS створює сучасний темний дизайн з яскравими зеленими акцентами, що виділяють важливі елементи. Темний фон і світлий текст забезпечують гарний контраст і зручність читання. Використовується простий шрифт, який сумісний з більшістю пристроїв.

Верхня навігація оформлена через *flex*-контейнер, що дозволяє рівномірно розташувати кнопки і поля вводу для зручного інтерфейсу. Кнопки мають плавні переходи кольору при наведенні, що додає динаміки і покращує взаємодію.

Слайдер із зображеннями працює за допомогою зміни прозорості, забезпечуючи плавне переливання картинок без ривків. Картки з елементами оформлені через grid, що робить макет адаптивним і впорядкованим. При наведенні на картки відбувається масштабування, що допомагає користувачу зосередитись на обраному елементі.

Форма для входу і реєстрації має прозорий фон із розмиттям, що робить її привабливою і сучасною. Поля вводу з округлими краями і виділеними межами при фокусі підвищують зручність користування.

Модальні вікна мають анімацію появи, що робить їх менш нав’язливими і плавними для сприйняття. Дизайн відповідає загальній темній палітрі з зеленими акцентами.

Стиль відповідає сучасним веб-трендам, зручний для користувачів і виглядає професійно. Структура *CSS* дозволяє легко розширювати і підтримувати код для подальшого розвитку сайту.

CSS потрібен для цього сайту, щоб зробити його привабливим, зручним і зрозумілим для користувачів. Без *CSS* сторінка була б просто текстом і зображеннями без стилю, що ускладнює сприйняття інформації. *CSS* дозволяє керувати кольорами, шрифтами, розміщенням елементів, анімаціями та адаптивністю сайту, завдяки чому користувачі можуть легко знаходити потрібне, комфортно взаємодіяти з інтерфейсом і отримувати приємний досвід під час використання.

На рисунку 3.26 зображено стилі заднього фону.

Рисунок 3.26 – Стилі заднього фону

Колір фону встановлений темним для комфортного перегляду, текст білим для контрасту. Вказані шрифти — сучасні та читабельні. Відсутні відступи навколо сторінки.

На рисунку 3.27 зображено стилістику верхньої навігації.

Рисунок 3.27 – Стилістика верхньої навігації

На рисунку 3.28 зображено масштабування зображень у слайдері.

Рисунок 3.28 – Масштабування зображень у слайдері

Зображення у слайдері займають весь контейнер, обрізаються за потреби. Розташовані абсолютним позиціюванням, щоб накладатися одне на одне. Спочатку прозорі, з плавною появою.

На рисунку 3.29 зображено поля введення даних.

Рисунок 3.29 – Поля введення даних

Поля введення у формі мають прозорий фон, округлі краї і центрований текст. Відсутність обводки під час фокусу для м’якшого вигляду.

На рисунку 3.30 зображено модальне вікно підтвердження.

Рисунок 3.30 – Модальне вікно підтвердження

Модальне вікно підтвердження має прозорий темний фон із розмиттям, округлі краї і яскраву зеленої рамку, що виділяє його на тлі.

Ці стилі створюють приємний сучасний вигляд сайту, забезпечують зручність і логічну структуру інтерфейсу.

3.3 Під’єднання коду з сервером та базою даних (*PHP*)

В цьому коді *PHP* :

Спочатку він встановлює, що відповідь буде у форматі JSON, щоб клієнт розумів, який формат отримує.

Потім зчитує дані, які прийшли в тілі запиту. Ці дані — це *JSON*, що містить

ім’я користувача та пароль. *PHP* декодує цей *JSON* у звичайний масив для зручної роботи.

Далі перевіряє, чи були надіслані ім’я користувача і пароль, щоб уникнути ситуації, коли ці поля порожні.

Потім відкриває файл, де зберігаються всі зареєстровані користувачі. Якщо такого файлу немає, створює порожній масив для збереження користувачів. Перевіряє, чи вже існує користувач із таким ім’ям, щоб не створити дубліката. Якщо користувача немає, додає нового користувача з його ім’ям і паролем (в цьому коді пароль зберігається у відкритому вигляді, але насправді краще його хешувати для безпеки).

Після цього записує оновлений список користувачів назад у файл, зберігаючи його у форматі *JSON*, щоб було легко читати і оновлювати.

І в кінці повертає клієнту відповідь — у форматі *JSON* — яка повідомляє, чи пройшла реєстрація успішно, чи сталася помилка (наприклад, користувач уже існує або не було передано необхідні дані).

Весь цей процес відбувається на сервері, тобто користувач не бачить код *PHP* і не може його змінити, а отримує тільки результат роботи у вигляді відповіді від сервера. Веб-сервер (наприклад, *Apache* або

На рисунку 3.31 зображено Файли *PHP*.

Рисунок 3.31 – Файли *PHP*

На рисунку 3.32 зображено Під’єднання до *JSON* файлу.

Рисунок 3.32 – приклад формату введення даних в базу

Коли користувач заповнює форму реєстрації на сайті і натискає кнопку відправки, браузер формує запит до сервера. Цей запит зазвичай відправляється методом *POST* і містить дані у форматі *JSON* — це зручний текстовий формат для передачі структурованої інформації.

На сервері *PHP* приймає цей запит. Спершу воно читає сирі дані, які надійшли у тілі запиту, використовуючи спеціальну функцію. Ця функція повертає строку, що містить *JSON*. Щоб працювати з цими даними, *PHP* розкодовує цей *JSON* у звичайний асоціативний масив — це внутрішній формат, з яким простіше оперувати.

Після цього код бере з масиву поля для логіна і пароля. Він перевіряє чи ці поля не порожні — якщо так, сервер повертає повідомлення про помилку і припиняє подальшу роботу. Така перевірка важлива, бо збереження порожніх даних неможливе і небажане.

Далі *PHP* працює з файлом, в якому зберігаються всі зареєстровані користувачі. Якщо файл не знайдено, то створюється порожній масив, який згодом буде містити інформацію про користувачів. Якщо файл існує, його вміст зчитується і розкодовується з *JSON* у масив.

Потім виконується перевірка, чи немає вже користувача з таким логіном. Якщо знайшли, сервер відправляє повідомлення про те, що користувач уже існує і припиняє виконання скрипта.

Якщо користувача з таким логіном немає, новий запис додається у масив користувачів — це асоціативний масив з ключами логін і пароль. Важливо пам’ятати, що зберігати паролі у відкритому вигляді не безпечно, і у реальних проектах паролі хешують перед збереженням.

Після додавання нового користувача весь масив знову кодується у *JSON*. При цьому використовується форматування для зручності читання і збереження кирилиці без проблем. Потім цей *JSON* записується у файл, оновлюючи його вміст.

3.4 Тест функціоналу сайту та аналіз отриманих даних

Коли користувач вводить логін і пароль у форму реєстрації і натискає кнопку, браузер відправляє ці дані у форматі *JSON* на серверний скрипт *PHP*. На рисунку 3.33 зображено внесення даних у форму реєстрації.

Рисунок 3.33 – Внесення даних при реєстрації у файл *JSON*

Потім коли дані записалися завдяки ним можна увійти на головну сторінку сайту

На рисунку 3.34 зображено вікно входу.

Рисунок 3.34 – вікно входу та введення даних

На рисунку 3.35 зображено головну сторінку сайту.

Рисунок 3.35 – головна сторінка сайту

Та при натисканні на кнопки вийти виводить вікно з текстом”ви впевнені що хочете вийти?” “Так\Ні”

На рисунку 3.36 зображено модальне вікно кнопки цікавіше та вийти.

Рисунок 3.36 – вікно кнопки цікавіше та вийти

І на останок виведення відео(при натисканні на іконку фільму або назву виводиться плеєр з програванням трейлеру)

На рисунку 3.37 зображено сторінку відтворення трейлеру.

Рисунок 3.37 – сторінка відтворення трейлеру

3.5 Аналіз отриманих результатів

В нас є сторінка логіну

На рисунку 3.38 зображено сторінку входу.

Рисунок 3.38 – Сторінка входу

На рисунку 3.39 зображено сторінку реєстрації.

Рисунок 3.39 – Сторінка реєстрації

На рисунку 3.40 зображено головну сторінку.

Рисунок 3.40 – Головна сторінка

На рисунку 3.41 зображено вікно виходу.

Рисунок 3.41 – Вікно виходу

На рисунку 3.42 зображено вікно цікавіших трейлерів.

Рисунок 3.42 – Вікно цікавіших трейлерів

На рисунку 3.43 зображено вікно пошуку.

Рисунок 3.43 – Вікно пошуку

3.6 Варіанти вдосконалень веб-застосунку

На моєму сайті я б удосконалив кілька важливих речей. Насамперед покращив би безпеку — замість зберігання паролів у відкритому вигляді використав би хешування, щоб жодна стороння особа не змогла просто так побачити ці дані. Також

додав би перевірку введених даних на сервері, не лише у формі, щоб уникнути некоректних або шкідливих запитів.

Ще одна важлива зміна — перехід із текстового файлу *JSON* на справжню базу даних. Це зробить сайт більш стабільним, швидким і масштабованим. Для зберігання користувачів, фільмів чи іншої інформації база буде надійнішою.

Із візуальної частини я б зробив сайт повністю адаптивним для телефонів і планшетів. Додав би трохи більше анімацій для приємного інтерфейсу. Наприклад, щоб кнопки м’яко реагували на натискання, а повідомлення про помилки або успіх виглядали більш сучасно — через модальні вікна або спливаючі підказки.

Ще одна ідея — зробити можливість входу для користувачів, щоб після реєстрації вони могли мати свій особистий кабінет. Там можна буде додавати улюблені фільми або зберігати переглянуті трейлери.

ВИСНОВКИ

У межах виконання кваліфікаційної роботи було створено інформаційний вебсайт, який дозволяє користувачам переглядати трейлери нових фільмів. Під час розробки проєкту були застосовані знання з основ вебпрограмування, а саме використано

HTML для структури сторінок, *CSS* для оформлення, *JavaScript* для додаткової взаємодії з користувачем, а також PHP для обробки форм реєстрації й авторизації. Зберігання даних користувачів реалізовано через *JSON*-файл у вигляді локальної бази.

Розроблений сайт має простий, зрозумілий інтерфейс, дає можливість переглядати відео, перемикатися між сторінками з трейлерами, реєструватися та входити в систему. Це дає змогу зручно та швидко знайомитися з новинками кіно у вигляді відеоанонсів.

Під час реалізації було виконано повний цикл створення вебзастосунку: від планування та проєктування до безпосереднього кодування й тестування. Проєкт охопив усі основні етапи життєвого циклу розробки: від визначення мети та структури ресурсу до безпосередньої реалізації функціоналу з використанням *HTML, CSS, JavaScript, PHP* та роботи з даними у форматі *JSON*.

У роботі було також враховано основи веббезпеки, забезпечено реєстрацію, авторизацію користувачів і взаємодію з локальною базою даних. Архітектура ресурсу побудована таким чином, щоб забезпечити простоту розширення функціоналу в майбутньому.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Теоретичні основи YouTube: веб-сайт Wikipedia. URL: https://uk.wikipedia.org/wiki/YouTube (дата звернення: 25.04.2025) 2. Теоретичні основи Vimeo: веб-сайт TorgSoft. URL: https://torgsoft.ua/articles/stati/business-info/vimeo/ (дата звернення: 25.04.2025) 3. Глобальне пояснення мов програмування: веб-сайт Wikipedia. URL: https://uk.wikipedia.org/wiki/Мова_програмування (дата звернення: 06.05.2025) 4. Вивчення структурної мови PHP: веб-сайт PHP.net. URL: https://www.php.net/manual/uk/introduction.php (дата звернення: 07.05.2025) 5. Мова функціоналу JavaScript: веб-сайт GoIT.Global. URL: https://goit.global/ua/articles/shcho-take-javascript-i-dlia-choho-vin-potriben/ (дата звернення: 08.05.2025)

6. Теорія мови стилю CSS: веб-сайт Mozilla.org. URL: https://developer.mozilla.org/ru/docs/Web/CSS (дата звернення: 09.05.2025) 7. Вивчення основ веб-інтерфейсу: веб-сайт. Wikipedia. URL: https://uk.wikipedia.org/wiki/Вебінтерфейс (дата звернення: 09.05.2025) 8. Вивчення сервера XAMPP: веб-сайт. Hostpro.ua. URL: https://hostpro.ua/wiki/ua/instructions/installing-and-configuring-the-local-xampp-web-server-on-windows/(дата звернення: 10.05.2025)

9. Теоретичний веб-Довідник з теорією CSS: веб-сайт. URL: https://css.in.ua/article/shcho-take-html_10 (дата звернення: 12.05.2025) 10. Підручник зі вступним поясненням JavaScript: веб-сайт. URL: https://uk.javascript.info/intro (дата звернення: 12.05.2025)

11. Мова програмування як набір інструкцій: веб-сайт. URL: https://www.mathros.net.ua/shho-take-mova-programuvannja.html (дата звернення: 12.05.2025)

12. Ключові переваги та методи застосування PHP: веб-сайт. URL: https://freehost.com.ua/ukr/faq/wiki/chto-takoe-php/ (дата звернення: 20.05.2025)

13. Структура сайту та основні правила їх розробки: веб-сайт. URL: https://webtune.com.ua/statti/web-rozrobka/struktura-sajtu/ (дата звернення: 25.05.2025) 14. Поняття UI або веб-інтерфейсу: веб-сайт. URL: https://terentevdesignstudio.com/blog/ui-abo-interfejs-koristuvacha/ (дата звернення: 05.06.2025)

15. Основні поняття бази даних: веб-сайт. URL: https://apeps.kpi.ua/shco-take basa-danykh\ (дата звернення: 05.06.2025)

КРИВОРІЗЬКИЙ ФАХОВИЙ КОЛЕДЖ
ДЕРЖАВНОГО НЕКОМЕРЦІЙНОГО ПІДПРИЄМСТВА
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

РЕЦЕНЗІЯ

на кваліфікаційну роботу

випускника спеціальності: 123 «Комп'ютерна інженерія»

відділення: комп'ютерної та програмної інженерії

циклова комісія: комп'ютерних систем та мереж

Ростислав СКРИПНИК
(ім'я, прізвище)

1. Актуальність теми: Обрана тема кваліфікаційної роботи «Інформаційний веб-ресурс для перегляду відеоновинок » є актуальною.

2. Кваліфікаційна робота відповідає темі, затвердженій наказом.

3. Завдання на виконання кваліфікаційної роботи виконано у повному обсязі.

4. В результаті виконання кваліфікаційної роботи було створено веб-ресурс для перегляду відеоновинок.

5. Якість виконання пояснювальної записки та ілюстративного (графічного) матеріалу відповідає вимогам Державних стандартів.

6. В кваліфікаційній роботі зроблений акцент на дані отримані на практиці («живі» експерименти).

7. Кваліфікаційна робота заслуговує оцінку «добре».

Рецензент _____
(науковий ступінь, посада)

«____»_____ 2025 р. _____ Тетяна РУБАН
(підпис) (ім'я, прізвище)

З рецензією ознайомлений _____ Ростислав СКРИПНИК
(підпис) (ім'я, прізвище)

КРИВОРІЗЬКИЙ ФАХОВИЙ КОЛЕДЖ
ДЕРЖАВНОГО НЕКОМЕРЦІЙНОГО ПІДПРИЄМСТВА
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

ВІДГУК
керівника кваліфікаційної роботи

випускника спеціальності: 123 «Комп'ютерна інженерія»

відділення: комп'ютерної та програмної інженерії

циклова комісія: комп'ютерних систем та мереж

Ростислав Скрипник
(ім'я, прізвище)

1. *Кваліфікаційна робота на тему «Інформаційний веб-ресурс для перегляду відео-новинок» виконана в ініціативному порядку.*
2. *Метою кваліфікаційної роботи є створення веб-застосунку для перегляду трейлерів.*
3. *Кваліфікаційна робота відповідає темі, затвердженій наказом начальника коледжу.*
4. *Кваліфікаційна робота виконана здобувачем освіти самостійно.*
5. *Здобувач освіти показав високі вміння роботи з літературними джерелами, аналіз теоретичного та практичного матеріалу, приймання обґрунтованих рішень, застосовування сучасних комп'ютерних інформаційних технологій.*
6. *Ростислав Скрипник показав достатній рівень дотримання вимог державних стандартів при виконанні кваліфікаційної роботи в цілому та оформленні пояснювальної записки .*
7. *Рівень виконаної кваліфікаційної роботи заслуговує оцінку «добре», відповідає набутих випускником знань, умінь та навичок, вимогам освітньої характеристики фахівця і можливість присвоєння йому кваліфікації фахівця освітнього ступеня «Фаховий Молодший Бакалавр» спеціальності 123 «Комп'ютерна інженерія».*

Керівник кваліфікаційної роботи

«06» 06 2025 р. _____ Олександр Гриценко
(підпис) (ім'я, прізвище)